scientific reports

Check for updates

OPEN

A quantitative benchmark of neural network feature selection methods for detecting nonlinear signals

Antoine Passemiers[1]✉, Pietro Folco[2], Daniele Raimondi[1,3]✉, Giovanni Birolo[2], Yves Moreau[1] & Piero Fariselli[2]✉

Classification and regression problems can be challenging when the relevant input features are diluted in noisy datasets, in particular when the sample size is limited. Traditional Feature Selection (FS) methods address this issue by relying on some assumptions such as the linear or additive relationship between features. Recently, a proliferation of Deep Learning (DL) models has emerged to tackle both FS and prediction at the same time, allowing non-linear modeling of the selected features. In this study, we systematically assess the performance of DL-based feature selection methods on synthetic datasets of varying complexity, and benchmark their efficacy in uncovering non-linear relationships between features. We also use the same settings to benchmark the reliability of gradient-based feature attribution techniques for Neural Networks (NNs), such as Saliency Maps (SM). A quantitative evaluation of the reliability of these approaches is currently missing. Our analysis indicates that even simple synthetic datasets can significantly challenge most of the DL-based FS and SM methods, while Random Forests, TreeShap, mRMR and LassoNet are the best performing FS methods. Our conclusion is that when quantifying the relevance of a few non linearly-entangled predictive features diluted in a large number of irrelevant noisy variables, DL-based FS and SM interpretation methods are still far from being reliable.

In the realm of modern Machine Learning (ML), the efficacy of deep neural networks has been nothing short of transformative, revolutionizing numerous domains like structural biology, image/speech recognition and Natural Language Processing[1–5]. However, this remarkable success is, to a large extent, contingent on the availability of substantial and informative data. When the volume of data is limited, particularly in scenarios where the ratio of number of instances n over number of features m is low, the performance of Deep Learning (DL) models can be significantly compromised. This *data underdetermination* is particularly common in Life Sciences, where Whole Exome or Genome Sequencing and other *omics*(e.g. Transcriptomics, Methylomics) can provide millions of measurements for each sample[6–8], while the number of samples cannot be arbitrarily increased, due to the experimental costs and sometimes even intrinsic population size limits (i.e., gathering patients with rare diseases). More commonly, such issues are referred to as *the curse of dimensionality* (or *Hughes phenomenon*)[9]. As the dimensionality of the problem increases, average performance can eventually increase, but will necessarily start deteriorating at a certain moment due to generalization issues.

Modern ML methods are therefore often placed in the undesirable position of having to deal with severely underdetermined datasets generated by underlying complex and highly non-linear mechanisms. Traditional Feature Selection (FS) methods are either designed primarily to detect linear or additive features (e.g., Lasso[10], Elastic net[11]) or designed based on heuristics (e.g., Relief[12], Random Forests[13]), and by construction they should therefore not be able to correctly identify the synergistic effects of highly non-linear features in high-dimensional space[14].

To address this limitation, several DL-based models for automatic FS have been recently introduced, with the idea of exploiting the predictive capabilities of NNs, such as CancelOut[15], DeepPINK[16], LassoNet[17], FSNet[18], Concrete Autoencoder[19]and Diet-Net[20]. Some of these methods are being adopted in different Life Sciences studies with apparent success[21–26]. These approaches seek to harness the inherent capabilities of NNs while simultaneously identifying and pruning irrelevant or redundant features, thus enhancing the model's efficiency and interpretability.

At the same time, also Saliency Maps (SM) and other gradient-based features attribution methods[27] have surfaced within this landscape as valuable tools for interpreting the internal decision process of NNs, that are

[1]ESAT-STADIUS, KU Leuven, Leuven, Belgium. [2]Department of Medical Sciences, University of Torino, Torino, Italy. [3] Institut de Génétique Moléculaire de Montpellier, Université de Montpellier, Montpellier, France. ✉email: antoine.passemiers@kuleuven.be; daniele.raimondi@igmm.cnrs.fr; piero.fariselli@unito.it

generally considered to be irremediably opaque *black boxes*. The most common approaches include Integrated Gradients[28], DeepLift[29], Input $\times$Gradient[30], SmoothGrad[31]and Guided Backpropagation[32]. They can be easily applied to any NN model with libraries such as Captum[33]. SM methods attempt to shed light on which features contribute most significantly to model predictions, offering insights into the network's decision-making process. However, they are often better suited for post-hoc analysis and lack the real-time adaptability needed for feature pruning during the learning phase. Moreover, some studies showing puzzling SM interpretation results and perplexing behaviors raised concerns about these methods[34–36].

Recent publications suggest that both DL-based FS and SM methods hold considerable potential and have demonstrated success in various applications[23,25,26,37]. However, a comprehensive *quantitative* assessment of their capabilities needs to be more conspicuously presented in the literature. In this study, we embark on a rigorous quantitative assessment of the potential of these DL-based FS approaches for detecting non-linear signals on synthetic benchmark datasets. Each of our datasets consists of a few variables that jointly and non-linearly correlate with the output class, as well as a variable number of irrelevant random features (decoys). By construction, these datasets cannot be segregated by linear decision boundaries, making linear FS methods totally unsuitable for the problem at hand.

Our results show that most of the tested DL-based FS and SM methods mostly fail at extracting relevant features even when relatively few decoy features are present. These findings indicate that the fields of DL-based FS and NNs interpretation have significant margins for improvement. Moreover, standardized *quantitative validation* benchmark datasets, such as the ones we propose here, should be used to assess their performance and limits, in order to give users a more realistic idea of the situations and extent in which these approaches are actually reliable.

This paper is structured as follows. In the Methods section, we present the characteristics and intuition behind the synthetic benchmark datasets we propose, as well as the real-world datasets used for benchmarking. We next describe the state-of-the-art FS and SM methods benchmarked in this study, dividing them in the "Feature Attribution", "Embedded FS" and "Filter FS" macro-categories. In the Results section, we present the benchmark results on each synthetic dataset, respectively, and summarize the results in a separate sub-section. The last sub-section is dedicated to the real datasets. Finally, we summarize the findings and limitations of our study in the Discussion section.

Methods

Building synthetic non-linearly separable datasets as a benchmark for FS methods

The goal of this study is to benchmark Deep Learning (DL)-based Feature Selection (FS) methods and gradient based Neural Networks (NNs) feature attribution methods such as SM, gauging their ability to distinguish noisy decoys from jointly and non-linearly relevant features. In order to quantitatively evaluate this, we designed 5 synthetic datasets, called RING, XOR, RING+OR, RING+XOR+SUM and DAG. They have been purposely constructed to pose different challenges for the FS methods in terms of the complexity and the peculiarities of the patterns that must be learnt. The use of synthetic data allowed us to know which features are truly relevant and which are just decoys, providing us with a *ground truth* on which our benchmark can be based.

Each of our synthetic datasets contains $n = 1000$ observations and $m = p + k$ features. p and k denote the number of predictive and irrelevant features, respectively. For RING, XOR, RING+OR and RING+XOR+SUM, features are uniformly distributed in the [0, 1] interval. Within each dataset, the prediction labels were assigned to data points according to a non-linear function of the p predictive features. All the remaining k features are effectively random with respect to the labels, and thus act as decoys when it comes to feature selection. In all datasets, the number of positives is equal to the number of negatives, to prevent any artifact due to class imbalance. In the following, we describe in detail the function used to build the predictive features for the 5 synthetic benchmark dataset. The datasets are also visually shown in Fig. 1.

The RING dataset has circular, non-linear boundaries

Recognizing the circular shape within the data can be a complex task, impossible for linear additive models, and hard to approximate for piecewise linear architectures (i.e., NNs with ReLU-like activations). In the RING dataset, positive labels are assigned to the points that form a bi-dimensional ring, defined by the features in positions $j \in \{0, 1\}$. The total number of predictive features is 2. Let Y be a binary random variable representing the ground-truth label of a data point. Points where assigned to the positive class ($\mathrm{Y} = 1$) when:

$$\left| \sqrt{(\mathrm{X}_0 - 0.5)^2 + (\mathrm{X}_1 - 0.5)^2} - 0.35 \right| \leq 0.1151, \tag{1}$$

where X_0 and X_1 are uniformly-distributed random variables representing the first and second predictive feature, respectively. The 2 features in RING do not linearly correlate with the labels, but are detectable by univariate (non-linear) FS approaches. This represents the majority of features encountered in real settings, as the marginal distribution of each of such features is rarely identical for the positive and the negative class. Indeed, for a given feature X_i, $p(\mathrm{X}_i|\mathrm{Y} = 1)$ most often differs from $p(\mathrm{X}_i|\mathrm{Y} = 0)$ due to the presence of a predictive signal, confounding effects, or random discrepancies caused by the low sample size.

The XOR dataset presents the archetypal non-linearly separable problem

This dataset presents an archetypal problem in ML in which the models are tasked to predict the "exclusive or" (XOR) interaction between two binary features. The XOR dataset requires that models are able to capture non-

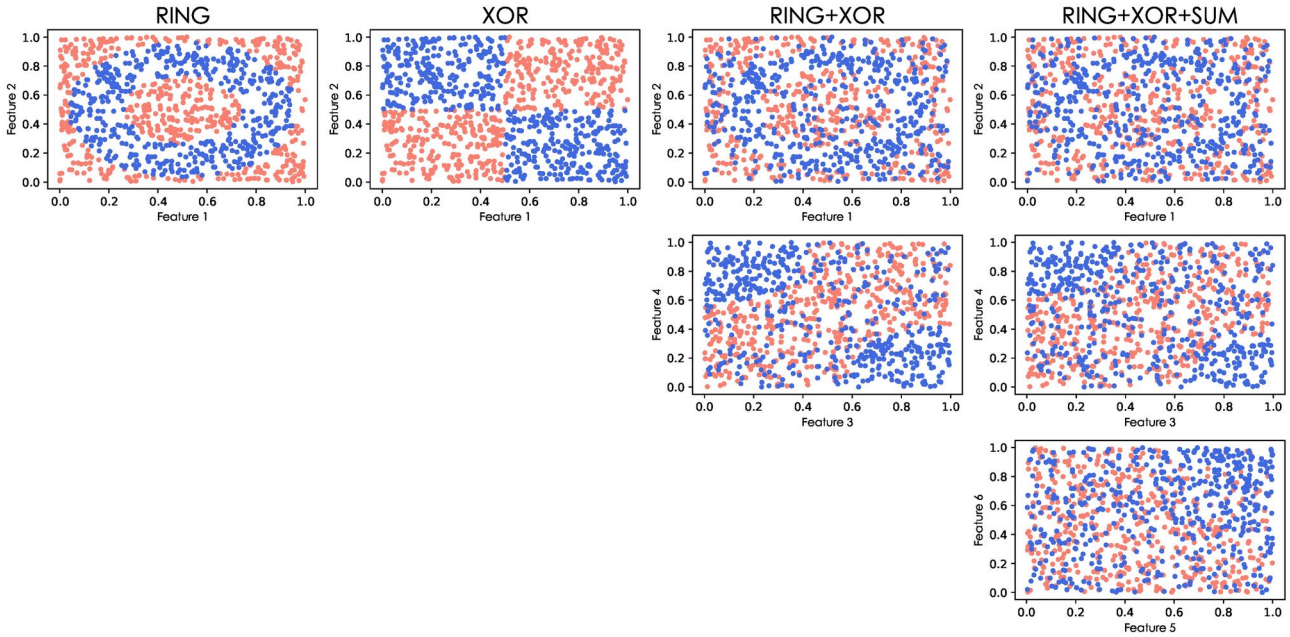

Fig. 1. The predictive features shown by pairs for each dataset. Orange and blue correspond to the positive and negative classes, respectively. Each column is associated with one dataset, and each row corresponds to a distinct pair of features.

linear synergistic relationships between input features, since the features taken singularly are uninformative. Contrary to the features in RING, these new features cannot be detected by univariate (non-linear) FS methods, as these features do not individually correlate with the labels. This poses an additional challenge, as the model is forced to consider the combination of the 2 features to be able to detect a predictive signal.

The bi-dimensional space formed by the features in position $j \in \{0, 1\}$ is divided into 4 equally-sized regions. Points in the upper left and lower right quadrants are labeled as positive samples, and remaining points as negative. The total number of predictive features is 2. Points were considered positive ($\mathrm{Y} = 1$) when:

$$(\mathrm{X}_0 - 0.5)(0.5 - \mathrm{X}_1) \geq 0. \tag{2}$$

The RING+XOR dataset combines RING and XOR features

To avoid privileging FS methods that consider small sets of candidate features during inference (e.g., Random Forest, mRMR), we increased the number of relevant features by merging the predictive features from RING and XOR. More formally, the samples that are either positive examples in the RING dataset (considering features X_0 and X_1) or positive examples in the XOR dataset (considering features X_2 and X_3) are positive. This dataset thus contains 4 predictive features, in positions $j \in \{0, 1, 2, 3\}$. Points were considered positive ($\mathrm{Y} = 1$) when they satisfied any of the following constraints:

$$\left| \sqrt{(\mathrm{X}_0 - 0.5)^2 + (\mathrm{X}_1 - 0.5)^2} - 0.35 \right| \leq 0.0704, \tag{3}$$

$$(\mathrm{X}_2 - 0.5)(0.5 - \mathrm{X}_3) \geq 0.0337, \tag{4}$$

where X_0, X_1 are RING predictive features, and X_2, X_3 are XOR predictive features.

The RING+XOR+SUM dataset adds linearly correlated features on top of previous ones

In real datasets, many features are likely to linearly correlate with the labels to some variable extent. While linear models may fail at uncovering the true underlying predictive pattern, the additive component can still be exploited to detect which features are relevant. Therefore, we acknowledged the prominence of additive features in real settings by incorporating 2 additive features to the existing ones in a non-linear fashion. More specifically, we introduced the inequality $\mathrm{X}_4 + \mathrm{X}_5 + \epsilon > 0.5$, where X_4 and X_5 are uniformly-distributed random variables representing features at positions $\{4, 5\}$, and $\epsilon \sim \mathcal{N}(\mu = 0, \sigma = 0.2)$ is a random noise term. Like in the RING and XOR datasets, the corresponding predictive features $\{0, 1, 2, 3\}$ do not contain noise. Points were considered positive ($\mathrm{Y} = 1$) when they satisfied at least one of the following inequalities:

$$\left| \sqrt{(\mathrm{X}_0 - 0.5)^2 + (\mathrm{X}_1 - 0.5)^2} - 0.35 \right| \leq 0.0479, \tag{5}$$

$$(\mathrm{X}_2 - 0.5)(0.5 - \mathrm{X}_3) \geq 0.0598, \tag{6}$$

$$\mathrm{X}_4 + \mathrm{X}_5 + \epsilon \geq 1.4074. \tag{7}$$

We purposely used different cutoffs on the RING features in Eqs. 1, 3 and 5 to prevent any class imbalance. Similarly, we chose different thresholds on the XOR features in Eqs. 2, 4 and 6. See Fig. 1 for a visual explanation of the different datasets presented so far.

DAG: a benchmark dataset based on a graphical model

While for sufficiently large values of m the datasets defined above are sufficiently challenging, we built an additional dataset (called DAG) characterized also by explicitly modeled confounding effects. Indeed, these effects occur in most real-life settings, and are very likely to *misguide* the different models towards learning predictive patterns from irrelevant associations. This dataset has been generated by a directed graphical model. The exact procedure is detailed in Suppl. Mat. 1.1. By construction, features can be categorized based on their degree of relevance:

- Features $\mathbf{X}_i$ that are highly relevant as they are causal for the target variable Y: $\mathbf{X}_i \rightarrow \ldots \rightarrow \mathbf{Y}$
- Features $\mathbf{X}_i$ that are weakly relevant as they correlate with the target variable only due to indirect effects, like in forks: $\mathbf{X}_i \leftarrow \ldots \leftarrow \mathbf{X}_j \rightarrow \ldots \rightarrow \mathbf{Y}$
- Irrelevant featuresBy design, we end up with a large (up to 81) number of predictive features, mimicking the typical situation where many features jointly but loosely correlate with the labels.

Addition of real data from existing benchmarks

Because the synthetic datasets designed in our study are purposely challenging and therefore might be too peculiar compared to reality, we also compared the different algorithms on 11 real datasets. The 5 first datasets date back from the NIPS 2003 Feature Selection Challenge[38], and have been specifically designed for FS tasks. These datasets are referred to as ARCENE, DEXTER, DOROTHEA, GISETTE and MADELON. The 6 remaining datasets have also been used previously[39], and consist of the MNIST, MNIST-Fashion, ISOLET, COIL-20, Mice Protein Dataset, and Smartphone Dataset for Human Activity Recognition (HAR). Most datasets are available in the UC Irvine Machine Learning Repository. Statistics about each dataset have been compiled in Suppl. Tab.4.

Procedure for the quantitative benchmark of Feature Selection methods

We assessed the reliability of FS methods on non-linear prediction tasks with an increasing degree of difficulty by incrementally diluting the relevant features in the uniformly-distributed synthetic datasets described previously. An exponential increase of the number k of decoy features is added in each run. The number m of total features is $m \in \{2, 4\} \cup K$ for XOR and RING datasets, $m \in \{4\} \cup K$ for RING+XOR and $m \in \{6\} \cup K$ for RING+XOR+SUM, where $K = \{8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. For each run, we set the number of samples to $n = 1000$. We also investigated the effect of decreasing n in Suppl. Mat. 1.4.

For each run and synthetic dataset, we assessed both the predictive performances of the models tested, and their FS ability. For each embedded FS method that relies on a predictive model, we evaluated the latter using a 6-fold cross-validation (CV). We computed the area under the receiver operating characteristic curve (AUROC) and area under the precision-recall curve (AUPRC) for each split, and averaged the results over the 6 folds. We also reported these metrics for a baseline NN without prior or posterior FS. In the paper, we simply refer to it as "Neural Network".

Second, we evaluated the ability of each FS algorithm to rank the predictive features higher than the decoys. More specifically, we quantified it as the percentage of predictive features among the highest p and $2p$ top-ranked features, where p corresponds to the number of truly predictive features in each dataset ($p = 2$ for XOR and RING, $p = 4$ for RING+XOR, $p = 6$ for RING+XOR+SUM, $p = 7$ or 81 in DAG depending on the definition used). We refer to them as best p and best $2p$ scores for short.

Feature annotations are not publicly available for the 5 datasets from the NIPS 2003 Feature Selection Challenge, therefore making it impossible to disentangle relevant from irrelevant features. However, the number of relevant features is available: $p = 7000$ for ARCENE, $p = 10000$ for DEXTER, $p = 50000$ for DOROTHEA, $p = 3500$ for GISETTE, and $p = 20$ for MADELON. Therefore, we first ranked the features using each FS method, kept the p top ones, and assessed the predictive performance by validating a Random Forest model on these p features. Validation was based on the validation set provided in each dataset. Because the disparities in cross-validation performance were found to be subtle, we combined the AUROC and AUPRC scores into a single value $\sqrt{AUROC \times AUPR}$, and ranked the methods according to that value. Methods producing the same score were assigned the same rank. For the 6 remaining real-world datasets, we performed similar comparisons, but assumed p to be half the total number of features.

To ensure fair comparison, and avoid assigning good performance to badly-designed FS methods (where feature importances are influenced by their position/indices in the data matrix), we randomly permuted the columns of the input data matrix before fitting each model, and permuted the features back after computing the feature importances. For feature attribution methods (e.g. SM), only the points from the held-out sets have been used to select features (we also discussed this choice in Suppl. Mat. 1.6). We computed the best p and best $2p$ scores for each split, and averaged them across the 6 folds.

Machine learning models

To ensure a fair comparison between DL-based FS methods, we tried to reuse the same NN architecture whenever possible. We implemented our NNs with Pytorch[40]. The data has been centered by replacing each input vector x_i by $2x_i - 1$, so each feature ranges between -1 and 1. The default model was a three-layer perceptron with LeakyReLU activation functions, with a 0.2 slope for negative values. The 2 hidden layers had 16 neurons each. Additionally, L2 regularisation on the parameters was used, with a regularisation parameter of 10^{-5}. The model was trained with the Adam optimizer[41] for up to 1000 epochs, with a learning rate of 0.005 and a batch size of 64. A scheduler guided the optimisation of each model by adjusting the learning rate over time, decreasing it by a 0.9 factor every time the total loss function stagnates for 10 epochs (with a cooldown of 5 epochs). Gaussian noise, with a standard deviation of 0.05, was added to the inputs in order to regularise the models further. In order to minimize overfitting risks and get the best out of each NN-based approach, we implemented an early stopping criterion. We saved the NN parameters at each epoch and kept the ones that minimize the loss function on a held-out set composed of 20% of the training data. The training was interrupted prematurely if the validation loss had not decreased during the last 5 epochs. The vanilla neural network was entirely based on this architecture. Only the downstream parts of DeepPINK and CancelOut were based on this architecture. The other NN-based FS methods were either based on a third-party library (e.g., Concrete Autoencoder) or a drastically different architecture (i.e., FSNet).

Also, all feature attribution methods are based on the architecture described above. However, due to the peculiarities of some embedded FS methods or the constraints of their implementation, these latter methods might build on subtle variants of this architecture. When relevant, these differences are explained in the next section.

Summary of the benchmarked feature selection methods

Here we summarize the FS methods we benchmarked in this study. They can be grouped in 3 categories: Feature Attribution, Embedded FS, and Filter FS methods. We describe them in details in the following. For the sake of comparison, we also included FS approaches that are not based on neural networks, namely Relief[12], mRMR[42], Random Forest[13]and TreeSHAP[43].

Feature attribution methods

Feature attribution methods propose an *a posteriori* interpretation of the method M by approximately reconstructing the decision process followed by M in order to produce the prediction y_i. Gradient-based interpretation methods for NNs like SM belong to this category. Given a trained NN model M, the forward pass $M(x_i)$ of sample x_i is computed, alongside with the gradient $\partial M(x_i)/\partial x_i$ of the target output $y_i = M(x_i)$ with respect to the input x_i. The gradient values identify which positions in x_i are the most important for the prediction. Indeed, because the gradient points towards the direction of steepest descent, the highest components of the gradient indicate which input variables require the least change to produce the largest variation in the output y_i. In this study, we used the Captum[33]library to benchmark the Integrated Gradients[28], Saliency[27], DeepLift[29], Input $\times$Gradient[30], SmoothGrad[31], Guided Backpropagation[32]. From the same library, we also benchmarked non-gradient-based interpretation methods like Deconvolution[44], Feature Ablation, Feature Permutation[45]and Shapley Value[46] interpretation approaches. For SmoothGrad, data was injected with random noise sampled from a zero-centered Gaussian distribution with 0.1 standard deviation (Captum's implementation of NoiseTunnel). The operation has been repeated 50 times per input vector. For the Integrated Gradients, DeepLift, Feature Ablation and Shapley Value Sampling methods, the 0 vector was supplied as baseline point. The baseline point is used for different purposes depending on the method. For example, the basepoint provided for the Integrated Gradients method defines the point from which to compute the integral and smooth the feature attribution vector. Because all these gradient-based methods only provide instance-level feature importances, we computed the overall feature importances as the average of absolute values of the instance-level importances. Because it was *a priori* unclear to us whether these instance-level scores should be computed on the training set or the validation set (with their difference explained by the generalization error), we compared the two settings in Suppl. Mat. 1.6.

Embedded feature selection methods

The second category of FS approaches are the embedded methods, which operate *at training time*, such as FSNet[18], Concrete Autoencoder[19], CancelOut[15], Random Forest[13], DeepPINK[16]and LassoNet[17].

FSNet's architecture is composed of a Selector and an Encoder network, which branches into a Classifier and a final sub-network comprising a Decoder and a Reconstruction layer. The Selector consists in a matrix multiplication operation, involving a low-rank matrix of shape $k \times 2p$ obtained from a weights predictor, followed by a softmax activation (after addition of Gumbel noise to the logits): $2p$ features are selected, and we computed both best p and best $2p$scores using the feature importances as proposed in[18]. The weights predictor of the Selector is composed of a fully-connected layer with no activation. We chose the identity function for both the Encoder and Decoder, as no further refinement of the input features was deemed necessary, and because it keeps the number of layers similar to the other NN-based approaches in our benchmark. The Reconstruction network reverses the dimensionality reduction enforced by the Selector network, and performs a matrix multiplication analogously. The corresponding low-rank matrix is predicted from a predictor module composed of a fully-connected layer with no activation. The classifier has the same architecture as described in the previous section, except that its input size is restricted to $2p$. The whole model was trained for 2000 epochs. 30 bins (latent size of 10) were used to compute the input frequencies necessary to predict the low-rank matrices.

CancelOut is composed of the shared architecture described in the previous section, preceded by a CancelOut layer, and was trained in the same manner. We experimented with 2 variants of the method: 1) with a Sigmoid

activation and CancelOut weights regularization (we denote the corresponding model by "CancelOut (sigmoid)" for short), and 2) with a Softmax activation layer and without regularization ("CancelOut (softmax)"). In the former case, we used a $\lambda_1 = 0.2$ coefficient for the variance and a $\lambda_2 = 0.1$ for the regularization term. Because L1 regularization encourages importances weights to converge to 0.5 (sigmoid(0) = 0.5), we replaced it by the sum of CancelOut weights, unlike the original implementation. Indeed, this choice of regularization better promotes sparsity among feature importances. In both cases, CancelOut weights were initialized with the same value $\beta = 1$. The model has been trained for up to 300 epochs, as the convergence of CancelOut weights requires longer time than the optimization of the classifier alone.

The LassoNet[17] architecture contained 32 hidden neurons and ReLU activation functions, as only the latter were available among activation functions in the lassonet Python package. We used the default hyper-parameters for training.

To evaluate the Concrete Autoencoder (CAE), we used the concrete-autoencoderPython package[19], and implemented the underlying classifier with Keras[47]. The autoencoder contained 3 layers, the first one preceded by random Gaussian noise injection (0.1 standard deviation) and the two other layers preceded by dropout layers (20% dropout). Each hidden layer was composed of 32 neurons. The CAE has been trained for 30 epochs, with initial temperature 10 and final temperature 0.01 and using a learning rate of 1e-4. The CAE has been trained twice, for selecting p and once for selecting $2p$ features respectively.

DeepPINK requires knockoff features, as described in the framework of Model-X Knockoff features designed by Candès et al.[48]. Details of Model-X Knockoff features are explained in Suppl. Mat. 1.2. We generated knockoff features in two different ways depending on the nature of the dataset. For the DAG dataset, we generated Gaussian knockoff features. For the remaining 4 datasets, knockoff features were generated by simply sampling from a uniform distribution. The DeepPINK architecture was made of the aforementioned shared architecture, but preceded by two locally-connected layers as described in[16]. The weights of the downstream part of the network were penalized by L1 regularization, using the recommended value of $0.05\sqrt{2\log(m)/n}$.

Random Forests[13] were grown with the scikit-learnPython package[49] and were composed of 500 trees each. We used the default hyper-parameters. We selected features from trained Random Forests using the widely-used impurity-based feature importance scores, as implemented in scikit-learn. More specifically, the importance of a feature is defined as the total reduction in Gini impurity caused by all node splits involving that feature, averaged across all trees in the forest. TreeSHAP[43] was based on the same Random Forest model, and quantified the importance of each feature with additive Shapley values. We used the shap Python package to extract feature importances from our scikit-learn Random Forest models.

Filter feature selection methods

The last category of methods that we considered is the set of Filter FS algorithms, such as Relief[12]or Minimum Redundancy Maximum Relevance (mRMR)[42]. These methods perform only FS and do not provide predictions, therefore no AUROC or AUPRC values are reported. mRMR relies on correlation metrics such as Mutual Information[50] for discrete features, or the F-statistic for continuous features. In order to discretize the features, we divided each feature into 20 equally-sized bins, which we deemed sufficiently thin for capturing the non-linear dependencies, and sufficiently large for robust estimation ($\sim$ 40 observations per bin).

We evaluated the original Relief[12] algorithm which was specifically designed for binary classification problems. At each iteration, a random data point is selected, and the closest data point from the same class ("near-hit") and from the other class ("near-miss") are identified. Then, the importance of each feature is decreased (increased) based the squared difference for that feature between current point and the near-hit (near-miss) point. We used the Euclidean distance as distance metric.

Finally, in order to have at least one univariate FS method in our benchmark, we included mutual information, which quantifies the non-linear correlation between the target variable and each input feature from an information-theoretic perspective. We used the mutual_info_classif function from the scikit-learn Python package.

Results

Feature Selection (FS) is widely used across many fields of science[6,7,51–53]. The goal of FS algorithms is to identify or rank features in function of the *predictive signal* they carry with respect to a prediction label. In this study, we benchmarked 20 FS methods on 5 synthetic datasets (RING, XOR, RING+XOR, RING+XOR+SUM and DAG) providing non-linear binary classification tasks (see Methods). These datasets represent classical ML problem, such as the XOR problem, the discrimination of points lying on a ring-shaped subspace and combinations thereof (see Methods for more details). The final results of our benchmark are summarized in Table 2. In the following sections we first discuss and analyze the results obtained within each dataset.

Random forests, mutual information and mRMR outperform other methods on RING by a large margin

In the RING dataset, positive labels are associated to the points lying on a bi-dimensional ring defined by features in positions $\{0, 1\}$ (see Suppl. Fig. S1).

Top panels in Fig. 2 show the AUROC and AUPRC of all trained models, as a function of the total number of features $m = p + k$. It is noteworthy that all models but the Random Forest have AUROCs and AUPRCs approaching a random predictor (50%) for values of m greater than 32. This effect is consistent with the percentages of relevant features reported in the bottom graphs of Fig. 2. Random forests, TreeSHAP and mRMR perfectly re-identified the relevant features, even when $m = 2048$. Mutual information, the only univariate FS method, perfectly re-identified relevant features up to $m = 1024$. However, the decaying predictive performance

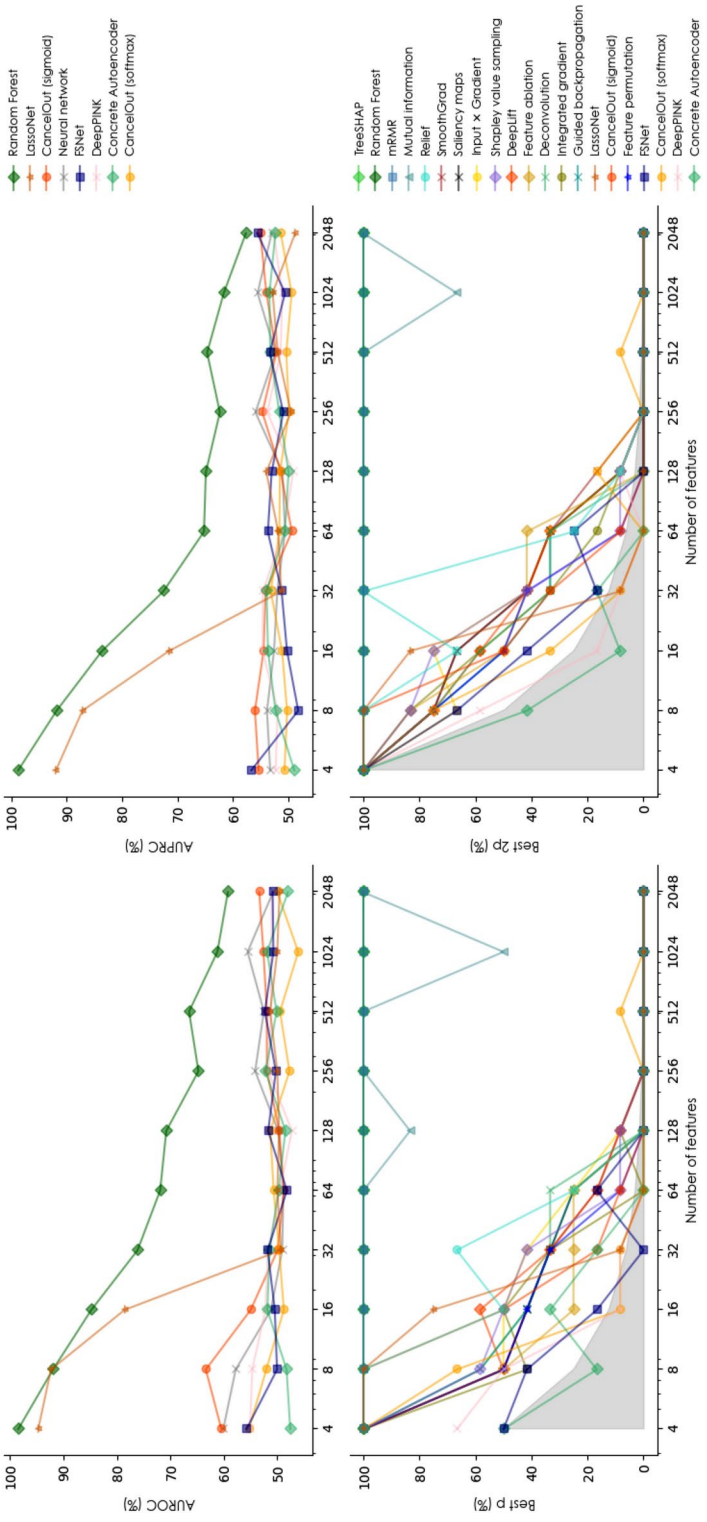

Fig. 2. Performance of the different models and feature selection methods on the RING dataset. (Top) AUROC and AUPRC of each trained model as a function of the number of features. (Bottom) Percentage of relevant features selected by each FS method in the top p and $2p$, respectively. Shaded areas correspond to the best p and best $2p$ scores of a dummy FS method that performs worse than random. Methods have been sorted by decreasing order of average performance in the legend.

of Random Forests (as m increases) suggests that tree-based models still loose their ability to optimally model the data.

Neural networks are better suited for solving the XOR problem

As shown in Fig. 3, NN models obtained overall better results, and mRMR underperformed, even for a small number of irrelevant features k. In particular, LassoNet reached maximal best p and best $2p$ scores for each $m \leq 512$, and $> 90\%$ AUROC and AUPRC for each $m \leq 512$. This relatively high predictive performance of LassoNet can be attributed to the internal cross-validation procedure used to find the optimal degree of sparsity of the NN parameters. The FS methods, besides LassoNet, that found the highest proportions of relevant features are Relief, CancelOut (sigmoid), Random Forests, and feature attribution methods such as Guided Backpropagation or Feature Permutation. Most methods underwent drastic performance losses when $m \geq 128$.

RF, mRMR and LassoNet are the best performing methods on RING+XOR

This dataset comprises four relevant features ($p = 4$). Similarly to what we observed for the RING dataset, we see from Fig. 4 and Tab. 2 that Random Forests and TreeSHAP (which interprets trained Random Forests) performed the best, along with mRMR. Random Forests achieved high AUROC and AUPRC values and correctly ranked the features in most settings. In particular, it is the only method that revealed capable of identifying over 80 % of relevant features (on average) when $m \leq 2048$. For any other method to achieve a similar percentage of relevant features, the total number of features needs to be decreased to $m = 16$, thus requiring to make the problem orders of magnitude simpler. Overall, only RF, mRMR and TreeSHAP result in best p and best $2p$ scores $> 60\%$ when $m > 64$. All remaining methods except LassoNet appear to perform close to random for $m \geq 256$. Indeed, LassoNet starts behaving as a random predictor only for $m \geq 1024$.

RING+XOR+SUM is challenging for all methods for both prediction and FS

There are six relevant features in this dataset, two of them being linear with some additive noise. The prediction problem appears to be quite difficult, with AUROC and AUPRC values < 0.85 for all methods even for $p = 6$ when no decoy feature is added (see Fig. 5). Overall performance seems to decrease more gradually with m than in other datasets. This can be explained by the slightly larger number of relevant features ($p = 6$) as well as their heterogeneity, allowing models to detect the features that are the easiest to pick for them (XOR features for NNs, RING features for mRMR and tree-based models, SUM features for most models). RF/TreeSHAP and mRMR are the best FS methods, with the best p scores $\geq 40\%$ and the best $2p$ scores $\geq 60\%$ for each value of m. FSNet ranked last as both predictive model and FS method. Finally, we observe that CancelOut with Sigmoid activation clearly outperformed its Softmax variant by a large margin.

Disentangling causal from spurious effects on DAG is too challenging for FS methods

In the fifth and last dataset in this benchmark, which has been generated by a graphical model (see Methods), relevant features can be defined either as being causal for the observed variable Y, or being predictive for Y due to indirect (confounding) effects. We refer to this dataset as DAG. In Table 1, we report the best p and $2p$

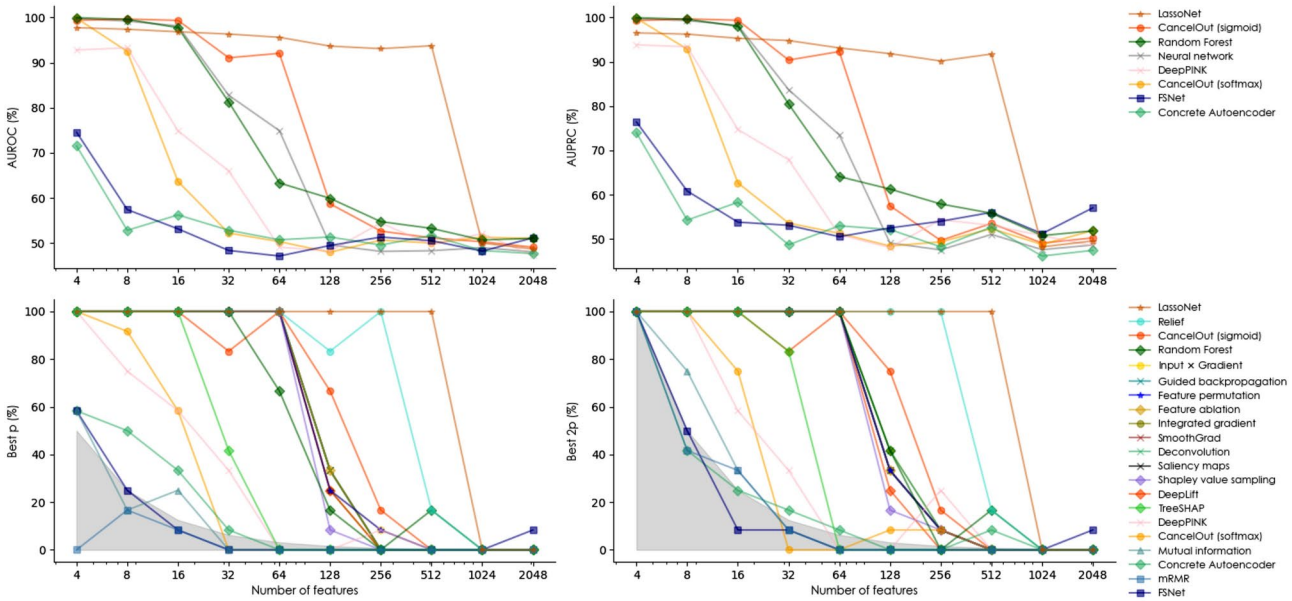

Fig. 3. Performance of the different models and feature selection methods on the XOR dataset. (Top) AUROC and AUPRC of each trained model as a function of the number of features. (Bottom) Percentage of relevant features selected by each FS method in the top p and $2p$, respectively. Shaded areas correspond to the best p and best $2p$ scores of a dummy FS method that performs worse than random. Methods have been sorted by decreasing order of average performance in the legend.

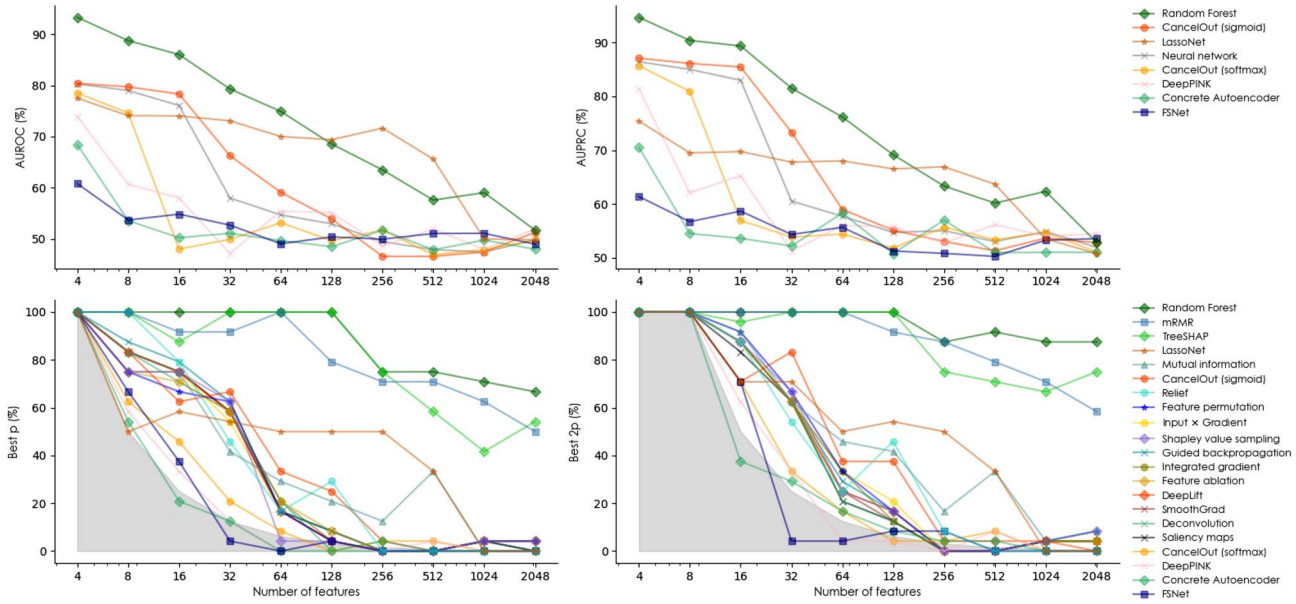

Fig. 4. Performance of the different models and feature selection methods on the RING+XOR dataset. (Top) AUROC and AUPRC of each trained model as a function of the number of features. (Bottom) Percentage of relevant features selected by each FS method in the top p and $2p$, respectively. Shaded areas correspond to the best p and best $2p$ scores of a dummy FS method that performs worse than random. Methods have been sorted by decreasing order of average performance in the legend.

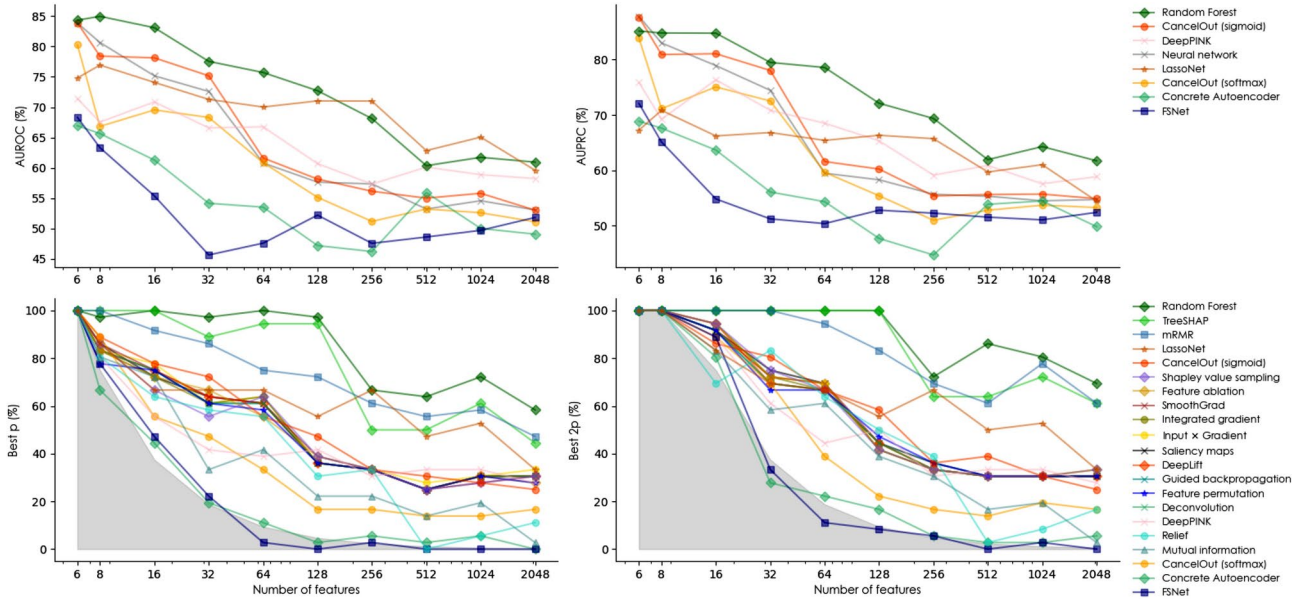

Fig. 5. Performance of the different models and feature selection methods on the RING+XOR+SUM dataset. (Top) AUROC and AUPRC of each trained model as a function of the number of features. (Bottom) Percentage of relevant features selected by each FS method in the top p and $2p$, respectively. Shaded areas correspond to the best p and best $2p$ scores of a dummy FS method that performs worse than random. Methods have been sorted by decreasing order of average performance in the legend.

scores based on these two definitions. We can see that, in both settings, TreeSHAP and its underlying Random Forest model are outperforming other methods. Concrete autoencoder, FSNet and CancelOut (softmax) failed at detecting relevant features. Feature attribution methods performed essentially the same, as also observed on the 4 other datasets. Overall, none of the benchmarked approaches seemed to be significantly better than the others at disentangling indirect correlations from causal effects on the DAG dataset.

Dataset	% causal features		% correlated features		Performance (%)	
Method	Best p	Best 2p	Best p	Best 2p	AUROC	AUPRC
Saliency maps	14.3	14.3	9.9	13.0	–	–
Integrated gradient	14.3	16.7	9.7	13.8	–	–
DeepLift	14.3	16.7	9.7	13.8	–	–
Input × Gradient	14.3	14.3	9.1	13.4	–	–
SmoothGrad	14.3	14.3	9.7	13.6	–	–
Guided backpropagation	14.3	14.3	9.9	13.0	–	–
Deconvolution	14.3	14.3	9.9	13.0	–	–
Feature ablation	14.3	14.3	9.1	13.6	–	–
Feature permutation	11.9	16.7	9.3	14.2	–	–
Shapley value sampling	14.3	16.7	9.1	13.6	–	–
Mutual information	14.3	14.3	7.0	10.3	–	–
mRMR	16.7	19.0	6.6	11.5	–	–
LassoNet	14.3	14.3	6.8	11.1	71.9	66.0
Relief	14.3	14.3	6.8	9.5	–	–
Concrete Autoencoder	2.4	11.9	4.9	8.0	50.1	52.8
FSNet	0.0	0.0	3.9	9.3	46.4	49.1
CancelOut (softmax)	9.5	23.8	7.0	12.1	51.7	52.8
CancelOut (sigmoid)	16.7	19.0	9.7	14.8	58.1	58.7
DeepPINK	14.3	14.3	8.4	12.3	70.4	69.0
Random Forest	21.4	40.5	12.8	**17.9**	**75.4**	**73.9**
TreeSHAP	**28.6**	**42.9**	**13.4**	17.3	–	–

Table 1. Best p and best 2p scores of all feature selection methods on the DAG dataset. $p = 7$ when considering only causal features as relevant (first multi-column) and $p = 81$ when also including confouders (second multi-column). Additionally, AUROC and AUPRC scores have been reported for embedded methods. Significant values are in bold.

Benchmark summary

To give the reader a summary of the results obtained by the benchmarked methods on all the datasets, we summarized them in Table 2. It shows the best p and best $2p$ scores for all FS methods on each dataset. For readability purposes, we only reported the mean scores, computed across all values of m. The table is divided in two parts: instance-level feature attribution methods (*a posteriori* gradient-based methods such as Saliency Maps) in the upper part, and the remaining approaches in the lower. The best performing methods from each category have been highlighted in bold. Among model-based and *a priori* FS techniques, TreeSHAP and its underlying model Random Forests both outperformed all approaches by a large margin on all datasets but XOR. Overall, mRMR appears to be the second best-performing technique, despite its underperformance on XOR. Contrary to mRMR, LassoNet achieved maximal performance on XOR, while getting average results on the remaining datasets. Instance-level *a posteriori* methods do not show significant differences.

In order to compare the methods in situations where the curse of dimensionality is exacerbated, we reported the same results on sub-sampled versions of the same datasets, with $n \in \{250, 500\}$. These results are shown in Supp. Tables 1 and 2. We observed that mRMR, tree-based methods and LassoNet consistently outperform other methods in the same settings.

Random forests outperform other methods on real-world datasets

Finally, we benchmarked the different algorithms on 11 real-world datasets where the number of features m can greatly exceed the number of training examples n, with the most extreme case being DOROTHEA ($n = 800$, $m = 100,000$). For each method and dataset, we computed an overall score as $\sqrt{\text{AUROC} \times \text{AUPRC}}$ and reported the results in Tab. 3. Most datasets contain a substantial number of relevant features, which contributes to overall high performance and results in only subtle differences among feature selection (FS) algorithms, particularly in the six datasets that did not originate from the NIPS 2003 Feature Selection Challenge.

To enhance clarity, we ranked the methods for each dataset, averaged these ranks, and presented this final value in the last column of Tab. 3. Random Forests emerged as the best performer, with an average rank of 6.273, while TreeSHAP struggled significantly on the COIL-20, HAR, and DOROTHEA datasets, yielding an average rank of 14.000. LassoNet ranked second (7.182), followed by Input × Gradient (7.545), Feature Ablation (7.636), and CancelOut (sigmoid) (7.818). Surprisingly, mRMR did not perform as well as it did on the synthetic datasets, placing among the lowest-ranked methods (14.364).

Average running time was the largest for FSNet (49884 seconds), which can be explained by our arbitrary choice to set the latent space size to the number of selected features, which is very large in some datasets (e.g., 50000 for DOROTHEA). Shapley value sampling was by far the most expensive feature attribution method (12076 seconds), as Captum performs 25 random feature permutations by default, therefore increasing the

Dataset	RING		XOR		RING+XOR		RING+XOR+SUM		DAG	
Method	Best k	Best 2k	Best k	Best 2k	Best k	Best 2k	Best k	Best 2k	Best k	Best 2k
Saliency maps	31.8	38.6	57.6	58.3	34.6	38.8	53.9	60.3	14.3	14.3
Integrated gradient	30.3	36.4	56.8	58.3	34.2	40.4	53.6	60.3	14.3	16.7
DeepLift	33.3	37.9	56.8	56.8	34.6	40.0	53.9	60.0	14.3	16.7
Input × Gradient	34.1	38.6	57.6	59.1	34.2	41.2	54.7	60.3	14.3	14.3
SmoothGrad	31.8	39.4	57.6	58.3	34.2	39.6	54.2	60.6	14.3	14.3
Guided backpropagation	32.6	36.4	57.6	59.1	35.8	40.4	53.6	60.0	14.3	14.3
Deconvolution	33.3	36.4	57.6	58.3	34.2	39.2	53.6	59.7	14.3	14.3
Feature ablation	29.5	37.9	57.6	58.3	34.2	40.0	53.3	60.6	14.3	14.3
Feature permutation	30.3	34.1	57.6	58.3	33.3	42.1	52.5	60.0	11.9	16.7
Shapley value sampling	33.3	37.9	55.3	56.8	32.9	40.8	52.8	60.6	14.3	16.7
Mutual information	92.6	96.3	18.2	28.8	39.6	50.4	40.8	51.1	14.3	14.3
mRMR	**100.0**	**100.0**	11.4	25.8	81.7	88.8	74.7	84.7	16.7	19.0
LassoNet	34.8	35.6	**81.8**	**81.8**	44.6	52.9	64.2	67.8	14.3	14.3
Relief	40.2	45.5	72.7	74.2	37.1	42.1	43.9	53.3	14.3	14.3
Concrete Autoencoder	19.7	24.2	22.7	27.3	19.2	30.0	25.8	36.4	2.4	11.9
FSNet	20.5	31.8	18.2	25.0	21.2	29.6	25.3	35.0	0.0	0.0
CancelOut (softmax)	26.5	31.1	31.8	35.6	24.6	33.8	40.3	48.6	9.5	23.8
CancelOut (sigmoid)	34.1	35.6	60.6	61.4	37.5	44.2	55.8	62.2	16.7	19.0
DeepPINK	21.2	26.5	34.1	37.9	21.2	31.2	48.3	56.9	14.3	14.3
Random Forest	**100.0**	**100.0**	54.5	59.8	**88.8**	**95.4**	**85.3**	**90.8**	21.4	40.5
TreeSHAP	**100.0**	**100.0**	40.2	43.9	81.7	88.3	78.3	86.1	**28.6**	**42.9**

Table 2. Best k and best 2k score percentages on the 5 datasets. For the first 4 datasets, scores have been averaged over $m \in \{2, 4, 6, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. Top and bottom parts of the table correspond to instance-level feature attribution and embedded/filter FS methods, respectively. Best performing methods are highlighted in bold.

computation costs by a 25 factor. Relief took 7220 seconds to run on average, which can be attributed to its poor scalability: the algorithm first starts with the computation of all pairwise distances between data points. Among the best performing FS methods, Random Forest and Input × Gradient required only 11 and 19 seconds on average, respectively.

Discussion

Machine Learning models can still fail when they are the most suitable

Random forests have largely outperformed other methods on the RING, RING+XOR, RING+XOR+SUM and DAG datasets, both as a FS technique and as predictive models. They also ranked first on the average of the 11 real-world datasets. However, the synthetic dataset where RFs perform comparatively worse is XOR, which consists of data points obeying a simple logical rule (Fig. 3).

The quality of its feature selection and prediction is not linked to the sophistication of the modeling *per se*, *but rather the optimality of the decision tree induction algorithm*. Indeed, scikit-learn's implementation is based on the heuristic CART algorithm[54], which is unlikely to infer the tree with the highest information gain and minimal number of nodes. In particular, in the XOR dataset, selecting one of the two relevant features as the first decision split is not sufficient, as it does not produce any change in the class proportions within the newly obtained hyper-parallelepipeds. Therefore, RFs are counter-intuitively better at growing from ring-shaped data since any split on one of the two corresponding features results in an increase of class purity (strictly positive information gain). On the XOR dataset, optimal inference would require a lookahead of 1 feature or bivariate decision splits. In conclusion, the relatively lower performance of RFs on the XOR dataset can mainly be attributed to the sub-optimality of its underlying tree induction algorithm. Similarly, neural networks are naturally well suited for solving the XOR problem, but we demonstrated that they rapidly start to underperform after adding a sufficient number of distractors to the pool of features. This failure can not be attributed to the NN architecture modeling, but rather the NN parameter optimization itself.

Univariate feature selection remains relevant in a high dimensionality context

Although the datasets have been constructed in a way that they are highly challenging for both linear and univariate FS methods, it must be noted that (non-linear) univariate filter methods remain highly relevant in some contexts. First, they are computationally more efficient (and embarrassingly parallel), making them competitive in a high-dimensional setting (e.g. Whole Genome Sequencing data). Second, they can still capture relevant features when they *individually* correlate with the explained variable in a non-linear fashion. This is shown by the large performance of mutual information (MI) on the RING dataset (as shown in Table 2). These results suggest that mutual information (MI), which is a non-linear tool, consistently detects the reduction of

Method	Avg time (s)	Peak mem (Mb)	$\sqrt{AUROC \times AUPR}$ score per dataset											Average rank
			ARCENE	MADELON	DEXTER	DOROTHEA	GISETTE	FASHION	ISOLET	MICE	HAR	MNIST	COIL20	
Concrete Autoencoder	2908	7646	0.866	0.712	0.979	-	0.995	0.952	-	0.998	0.982	-	0.999	15.818
FSNet	49884	991	0.889	0.761	0.906	-	0.994	0.944	0.967	0.998	0.967	0.992	1.000	15.545
mRMR	9423	2196	0.873	0.643	-	-	0.995	0.958	0.975	1.000	0.981	0.994	1.000	14.364
TreeSHAP	730	5806	0.893	0.965	0.978	0.219	0.995	-	0.989	0.970	0.822	-	0.272	14.000
DeepPINK	7697	9052	0.877	0.755	0.972	-	0.995	0.962	0.989	0.999	0.983	0.996	0.999	13.000
CancelOut (softmax)	723	3430	0.881	0.742	0.951	0.824	0.994	0.961	0.986	0.999	0.984	0.997	1.000	12.727
Deconvolution	1	3618	0.870	0.613	0.979	0.813	0.995	0.962	0.989	0.999	0.984	-	0.999	11.909
Guided backpropagation	842	6520	0.870	0.613	0.979	0.813	0.995	0.962	0.989	0.999	0.983	0.997	-	11.818
Feature permutation	499	3317	0.887	0.606	0.981	0.815	0.995	0.961	0.988	0.999	0.983	0.997	0.999	11.364
Saliency maps	4	5167	0.870	0.613	0.979	0.813	0.995	0.962	0.989	0.999	0.983	0.997	1.000	11.000
DeepLift	4	3267	0.877	0.618	0.980	0.841	0.995	0.962	0.989	0.999	0.983	0.997	0.999	10.818
Integrated gradient	375	3263	0.890	0.615	0.975	0.830	0.995	0.962	0.988	1.000	0.984	0.997	0.999	10.727
SmoothGrad	202	3619	0.885	0.622	0.977	0.827	0.995	0.962	0.989	0.999	0.983	0.997	-	10.000
Mutual information	148	2569	0.876	0.832	0.981	0.836	0.995	0.960	0.987	1.000	0.982	0.997	1.000	9.818
Relief	7220	8765	0.884	0.943	0.972	-	0.995	0.958	0.989	1.000	0.984	0.997	0.999	9.455
Shapley value sampling	12076	3263	0.882	0.620	0.979	0.835	0.995	0.962	0.988	0.999	0.984	0.997	1.000	8.727
CancelOut (sigmoid)	6025	3267	0.877	0.697	0.977	0.843	0.995	0.963	0.989	0.999	0.983	0.997	1.000	7.818
Feature ablation	350	5953	0.879	0.651	0.981	0.833	0.995	0.962	0.988	0.999	0.984	0.997	1.000	7.636
Input × Gradient	19	3262	0.888	0.653	0.981	0.840	0.995	0.962	0.989	0.999	0.984	0.997	0.999	7.545
LassoNet	6843	5682	0.879	0.950	0.977	0.836	0.995	0.962	0.989	0.999	0.986	0.997	0.999	7.182
Random Forest	11	5682	0.899	0.965	0.975	0.840	0.995	0.960	0.990	0.999	0.986	0.997	1.000	**6.273**

Table 3. Computational resources required by each method, and performance on each of the 11 real datasets. The ranks of each method, averaged across the 11 datasets, have been reported in the last column. Rows have been sorted by decreasing average rank. Methods running longer than 3 days on a given dataset have been interrupted, marked with "-" in the table, and assigned the lowest ranks for that particular dataset.

entropy caused by the ring-shaped function that generated the data. Indeed, this ring-shaped data is leaking information about the class at the level of individual features (the probability distribution of each feature is altered when conditioned on the class distribution). This ability to capture information in a univariate fashion is also evidenced by the overall low rank of MI on the real-workd datasets (Table 3), showing that it performs comparately better than more sophisticated approaches such as DeepPINK or FSNet.

Even the simplest algorithm can be affected by *the curse of dimensionality*

Despite Relief being a simplistic and interpretable algorithm, it did not prove robust in high-dimensional settings, as extensively shown in our results on synthetic datasets. Relief is a multivariate and non-linear FS approach that iteratively selects data point pairs based on their distance. However, any method based on classical distance metrics (e.g., Euclidean, Manhattan) in high-dimensional settings will be affected by the *curse of dimensionality*. This problem is reflected in the fact that the higher the number of features, the less meaningful these distance metrics, as all data points start being highly and equally distant from each other: "most of the volume of a high-dimensional orange is in the skin, not the pulp"[55]. To illustrate this phenomenon on the XOR dataset, we reported the distributions of intra-class and inter-class pairwise Euclidean distances in Fig. 6. It appears that $m = 256$ features are sufficient to make the two distributions similar (p-value=0.045 > 0.01). Therefore, problem-specific distance metrics need to be designed in order to handle the presence of noise in high-dimensional spaces.

Are neural networks bad performers?

Given the various DL approaches we benchmarked in this paper, it is hard to formalize prior hypotheses explaining their behaviour. Nevertheless, in the following, we relate the inner workings of these models in the light of the results summarized in Table 2. To do so, in Fig. 7, we compare the training accuracy of our vanilla NN and the embedded NN-based FS methods on the XOR dataset and a purely randomly generated dataset without any feature correlating with the target variable. Strikingly, only 64 features are sufficient to make the NN overfit the 1000 points in the dataset when only decoy features are present.

This phenomenon can be attributed to the underdetermination of the problem: the more features, the more likely the model finds a non-linear combination of features that perfectly shatters the data (separates the data points based on the classes) just by random chance. This concept relates to the Vapnik-Chervonenkis (VC)[56] dimension, defined as the maximum number of points that can be shattered by a given binary classifier for *any* assignment of the class to the points. For piecewise linear NN architectures (i.e., traditional NNs with ReLU-like activation functions), of which our baseline NN is an instance, the lower bound on the VC-dimension is $\Omega\left(WL\log(W/L)\right)$, where W is the number of NN parameters, L the number of layers and $\Omega$the asymptotic notation for the lower bound[57]. Given the total number of features m, we obtain $W = 16m + 16 + 16^2 + 16 + 16 + 1$ weights for our vanilla NN. Therefore, the overfitting risks for piecewise linear NNs grow significantly faster than in more traditional ML models such as linear models, where the VC-dimension is at most $m + 1$.

As suggested in[58], the number of features should not exceed $\sqrt{n}$ when relevant features strongly correlate. This optimal number of features $\sqrt{1000} \simeq 31.2$ indeed corresponds to the m value at which the vanilla NN suddenly starts severely overfitting the training data even in the absence of relevant features ($m = 32$), as shown in Fig. 7.

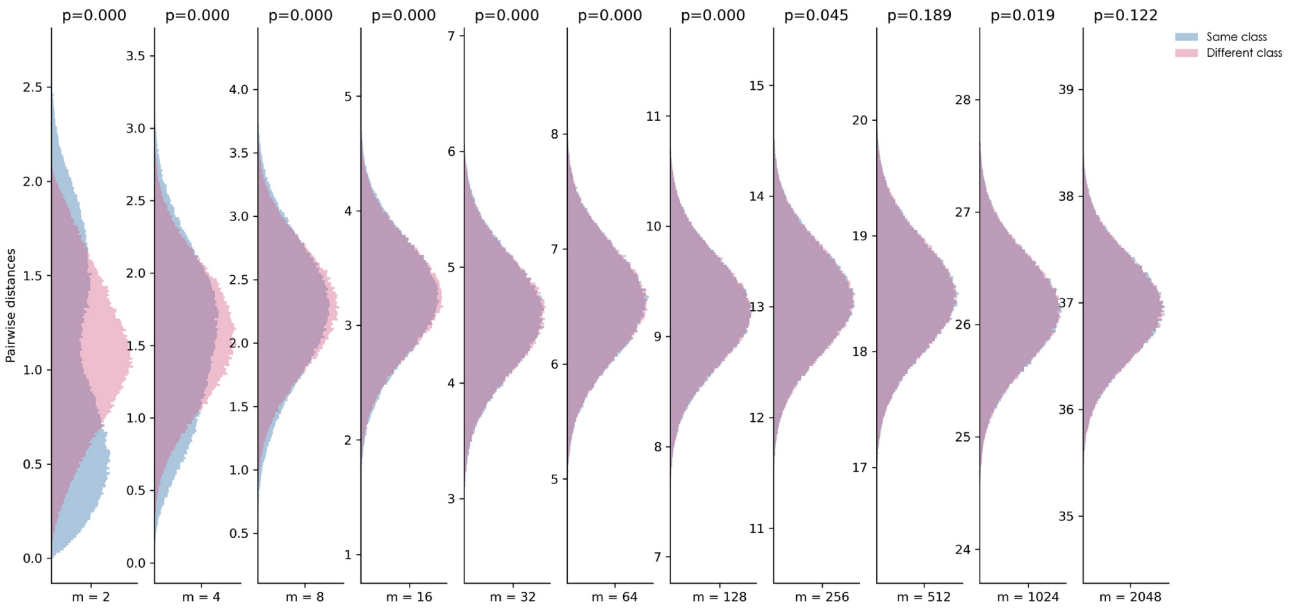

Fig. 6. Distribution of pairwise distances between points from the same class (blue) and from different classes (red) in the XOR dataset, for different numbers of features m. The p-value of a two-sample Kolmogorov-Smirnov test is reported for each m.

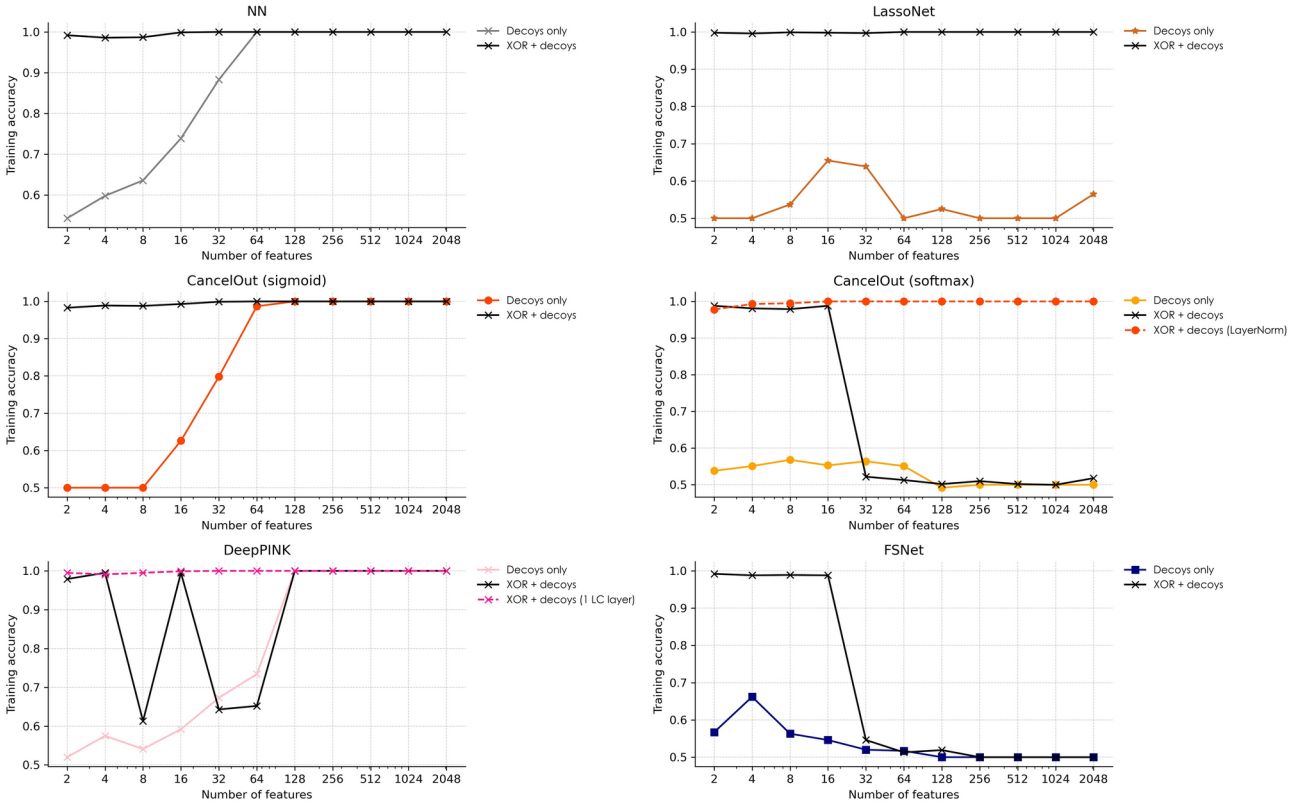

Fig. 7. Training accuracy on the XOR dataset ("XOR + decoy") and on a dataset only composed of m randomly and uniformly generated decoy features ("decoys only"), respectively. For this experiment, we removed the early stopping criterion to assess the model's ability to shatter the data. Additionally, we reported the performance on decoys for modified versions of CancelOut (softmax) and DeepPINK with dashed lines. For CancelOut (softmax), we added a Layer Normalization function between the CancelOut layer and the downstream NN. For DeepPINK, we removed the second locally-connected layer.

LassoNet is the only model that does not fit the training data when no relevant feature is present, even at $m = 2048$ (Fig. 7). This effect is likely due to LassoNet's internal cross-validation to find the sparsity pattern that minimizes generalization error. Its architecture contains a residual connection between the inputs and the output, therefore turning the network into the sum of a non-linear model $f_W(x)$ parameterized by W and a linear model $\theta^T x$ parameterized by θ. Therefore, the prediction function of the corresponding classifier is $\hat{y} = \sigma(\theta^T x + f_W(x))$, where σ is any activation function. LassoNet's ability to capture additive features explains its overall performance gain on the RING+XOR+SUM dataset compared to RING+XOR, as reported in Table 2.

CancelOut, with sigmoid activation, is only marginally better than the vanilla NN, showing that the CancelOut layer acts as a regularizer on the downstream network.

CancelOut with softmax activation does not overfit the decoy-only set and performs significantly better with layer normalization (Fig. 7(dashed orange line)). However, we omitted Layer Normalization in the architecture for benchmarking, as rescaling the features would invalidate the inferred Cancelout parameters and make the feature importances non-interpretable. Let us note that the distributional shift introduced by softmax activation was not mentioned in the original publication[15].

DeepPINK starts to perfectly overfit the decoy-only dataset when m is equal to 128. Furthermore, it produced inconsistent scores, demonstrating the training procedure's lack of reliability. The redundancy between its two locally-connected (LC) layers could be the cause, as it could make the optimization problem underdetermined. This is shown in Fig. 7 (dashed pink line), where removing the second LC layer stabilized the training, allowing the network to fit the XOR+decoy features for any m. As for CancelOut, we did not include this modified version in our benchmarking, as it was unclear how to compute feature importances without the second LC layer.

FSNet rightfully does not overfit the decoy-only data set. However, as the total number of features grows large enough ($m \geq 32$), it fails at fitting the XOR+decoy data. As in our other experiments, we set up FSNet's Selector layer with $2p$ output features. In the present case, $2p = 4$. Encoding the input samples into 4 features will likely cause an information bottleneck and, therefore, loose the predictive signal.

From a broader perspective, multiple mechanisms could be put in place to improve NN performance, such as bootstrapping or recursive feature elimination (RFE). Indeed, RFE can simplify FS by gradually removing features and reducing the underdetermination of the problem. On the other hand, bootstrapping has the potential to alleviate random fluctuations in Saliency Maps or neural network parameters by averaging across multiple runs. However, given the relatively demanding computational requirements of NN-based methods, we

did not implement any of these mechanisms as they would increase computational costs by several orders of magnitude.

Neural network hyper-parameter optimization can not overcome the Hughes phenomenon

While we originally selected the neural network hyper-parameters in such a way that NNs perform well on all synthetic datasets using the *same* hyper-parameter values, we did not over-tune them on each separate dataset and each m value separately, as it would result in excessive computational costs. On the real-world datasets, hyper-parameter optimization (HPO) of a dense multi-layer perceptron would become intractable. Moreover, excessive HPO of NNs would result in an unfair comparison with other methods, such as Random Forests, for which we used scikit-learn's default hyper-parameters (except n_estimators=500).

Nonetheless, we investigated whether the low performance of NNs on the synthetic datasets with large m was not directly due to the choice of hyper-parameters. The Optuna[59]Python package was used to perform HPO of the activation function, number of layers, number of hidden neurons per layer, standard deviation of the injected Gaussian noise, dropout rate, use of LayerNorm normalization, learning rate, batch size, L2 regularization rate (weight decay), optimizer, and whether to use (Adaptive) Sharpness-Aware Minimization[60,61]. Walltime was set to 12 hours for each dataset. The best hyper-parameters have been listed in Suppl. Tab.5 for $m = 128$, and Suppl. Tab.6 for $m = 2048$. We observed both the inconsistency of the results between datasets, but also the very low predictive performance overall (0.570 AUROC on RING, 0.576 AUROC on XOR, etc.). We conclude that HPO alone is not sufficient for alleviating the underdetermination issues, as more sophisticated approaches are needed to prevent NNs from getting distracted by decoy features.

Study limitations

Care should be taken to interpret the results presented in our study. The success of FS methods relies on the adequacy of their underlying model with the data, and due to the complexity of real datasets, it is practically impossible to determine whether a particular NN architecture is suitable or not. Also, the performance of two different NNs will necessarily be identical (under certain conditions) when averaged over all possible problems, as suggested by the No-Free-Lunch theorem[62], therefore making NN design and hyper-parameter optimization ill-defined when based on a single dataset. By cross-validating our NN models, using an early stopping criterion and restoring the NN parameters that minimize cross-validation error, while *not*overly hyper-optimizing the NNs towards each dataset, we proposed a realistic tradeoff and simulated the typical situation where the practitioner does not fully explore the NN solution space, or lacks the amount of data necessary to assess the generalization error, but uses the currently best ML practices at their disposal to mitigate these issues. Let us note that achieving the best generalization is easier when the models have sufficient insights about the data, for example by predefining the sparsity pattern of the NN weight matrices based on prior knowledge (e.g., known biological interactions)[63], by grouping the features based on their peculiarities (e.g., Group Lasso/LassoNet)[17]when regularizing, or by providing good initial values for the NN parameters (e.g., transfer learning[64]). We acknowledge that the results reported in this study might be dataset-dependent, and we tried to alleviate this effect by proposing 5 different synthetic datasets with various properties of the features that could be representative of several real-life cases.

Despite these limitations, our results suggest that in real-life scenarios where we do not know the optimal NN hyper-parameters and architecture for each specific dataset, the current DL-based FS methods have a very low probability of discovering relevant features, especially when the number of observations is of the same order of magnitude (or lower) as the total number of features. We also highlight the higher robustness of traditional FS methods compared to NNs in detecting additive features and features that individually and non-linearly correlate with the labels.

Conclusion

In this paper, we investigated the usefulness of non-linear FS approaches in the case of high-dimensional data with a low sample-to-feature ratio and non-linear data patterns. What emerges from the results is that Random Forests and mRMR outperformed neural network-based approaches on all synthetic datasets that exhibit non-linear correlations between individual features and the target variable, apart from the XOR dataset, where LassoNet was leading. Therefore, DNN models might not be the best choice for feature selection on datasets with these characteristics, i.e. a low density of relevant features, a prevalence of non-linear patterns and variance homogeneity (for both predictive and decoy features). Random Forests and LassoNet also led on the real-world datasets, while mRMR performed significantly worse. In real-life applications, we mainly encounter that relationships among the features are additive to some extent. However, our study indicates that when both additive and non-linearly entangled features are present, the latter are hardly detectable compared to the former unless a considerable number of samples are available to mitigate the underdetermination of the problem. In the light of the presented results, we suggest that the development of novel NN-based FS methods should be directed toward the integration of simple metrics such as mutual information to guide the learning and discourage NNs from relying on spurious correlations. Given the combinatorial nature of the FS problem, each of the presented methods could also be combined with heuristic search strategies such as Recursive Feature Elimination (RFE) for improved performance. However, many of them (e.g., FSNet, LassoNet) would necessitate methodological improvements to alleviate their rather large computational costs.

Data availability

The datasets used in the study can be generated using the code available at: https://github.com/AntoinePassemiers/Feature-Selection-Benchmark.

Received: 14 August 2024; Accepted: 6 December 2024
Published online: 28 December 2024

References

1. Jumper, J. et al. Highly accurate protein structure prediction with alphafold. *Nature* **596**, 583–589 (2021).
2. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *nature* **518**, 529–533 (2015).
3. Graves, A. et al. Hybrid computing using a neural network with dynamic external memory. *Nature* **538**, 471–476 (2016).
4. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
5. Van Dis, E. A., Bollen, J., Zuidema, W., van Rooij, R. & Bockting, C. L. Chatgpt: five priorities for research. *Nature* **614**, 224–226 (2023).
6. Nicora, G., Vitali, F., Dagliati, A., Geifman, N. & Bellazzi, R. Integrated multi-omics analyses in oncology: a review of machine learning methods and tools. *Frontiers in oncology* **10**, 1030 (2020).
7. Reel, P. S., Reel, S., Pearson, E., Trucco, E. & Jefferson, E. Using machine learning approaches for multi-omics data analysis: A review. *Biotechnology Advances* **49**, 107739 (2021).
8. Raimondi, D. *et al.* An interpretable low-complexity machine learning framework for robust exome-based in-silico diagnosis of crohn's disease patients. *NAR genomics and bioinformatics* **2**, lqaa011 (2020).
9. Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory* **14**, 55–63 (1968).
10. Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **58**, 267–288 (1996).
11. Zou, H. & Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **67**, 301–320 (2005).
12. Rosario, S. F. & Thangadurai, K. Relief: feature selection approach. *International journal of innovative research and development* **4** (2015).
13. Breiman, L. *Random forests. Machine learning* **45**, 5–32 (2001).
14. Hastie, T., Tibshirani, R. & Friedman, J. *The elements of statistical learning: data mining, inference and prediction* (Springer, 2009), 2 edn.
15. Borisov, V., Haug, J. & Kasneci, G. Cancelout: A layer for feature selection in deep neural networks. In *International Conference on Artificial Neural Networks*, 72–83 (Springer, 2019).
16. Lu, Y. Y., Fan, Y., Lv, J. & Noble, W. S. Deeppink: reproducible feature selection in deep neural networks. arXiv preprint arXiv:1809.01185 (2018).
17. Lemhadri, I., Ruan, F., Abraham, L. & Tibshirani, R. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research* **22**, 1–29 (2021).
18. Singh, D., Climente-González, H., Petrovich, M., Kawakami, E. & Yamada, M. Fsnet: Feature selection network on high-dimensional biological data. arXiv preprint arXiv:2001.08322 (2020).
19. Abid, A., Balin, M. F. & Zou, J. Concrete autoencoders for differentiable feature selection and reconstruction. arXiv preprint arXiv:1901.09346 (2019).
20. Romero, A. *et al.* Diet networks: Thin parameters for fat genomics. In *International Conference on Learning Representations* (2017).
21. Read, D. F., Lu, Y. Y., Cook, K., Roch, K. L. & Noble, W. S. Predicting gene expression in the human malaria parasite plasmodium falciparum. *bioRxiv* 431049 (2018).
22. Kassani, P. H., Lu, F., Le Guen, Y., Belloy, M. E. & He, Z. Deep neural networks with controlled variable selection for the identification of putative causal genetic variants. *Nature Machine Intelligence* **4**, 761–771 (2022).
23. Sarwar, N., Gregory, W., Kevrekidis, G. A., Villar, S. & Dumitrascu, B. Markermap: nonlinear marker selection for single-cell studies. arXiv preprint arXiv:2207.14106 (2022).
24. Fan, Z. *et al.* Deep neural networks with knockoff features identify nonlinear causal relations and estimate effect sizes in complex biological systems. *GigaScience* **12**, giad044 (2023).
25. Lapierre, L. R., Ritambhara, S. *et al.* A pan-tissue dna-methylation epigenetic clock based on deep learning. *NPJ Aging and Mechanisms of Disease* **8** (2022).
26. Sajwani, H. M. & Feng, S. F. Identifying snp associations and predicting disease risk from genome-wide association studies using lassonet. *bioRxiv* 2021–08 (2021).
27. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013).
28. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. *CoRR* (2017). arXiv:1703.01365.
29. Shrikumar, A., Greenside, P. & Kundaje, A. Learning important features through propagating activation differences. *CoRR* **abs/1704.02685** (2017). arXiv:1704.02685.
30. Shrikumar, A., Greenside, P., Shcherbina, A. & Kundaje, A. Not just a black box: Learning important features through propagating activation differences. *CoRR* **abs/1605.01713** (2016). arXiv:1605.01713.
31. Smilkov, D., Thorat, N., Kim, B., Viégas, F. B. & Wattenberg, M. Smoothgrad: removing noise by adding noise. *CoRR* **abs/1706.03825** (2017). arXiv:1706.03825.
32. Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014).
33. Kokhlikyan, N. *et al.* Pytorch captum. https://github.com/pytorch/captum (2019).
34. Adebayo, J. *et al.* Sanity checks for saliency maps. *Advances in neural information processing systems* **31** (2018).
35. Alqaraawi, A., Schuessler, M., Weiß, P., Costanza, E. & Berthouze, N. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, 275–285 (2020).
36. Nie, W., Zhang, Y. & Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, 3809–3818 (PMLR, 2018).
37. Feng, J., Zhang, H. & Li, F. Investigating the relevance of major signaling pathways in cancer survival using a biologically meaningful deep learning model. *BMC bioinformatics* **22**, 1–13 (2021).
38. Guyon, I., Gunn, S., Ben-Hur, A. & Dror, G. Result analysis of the nips 2003 feature selection challenge. *Advances in neural information processing systems* **17** (2004).
39. Balın, M. F., Abid, A. & Zou, J. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International conference on machine learning*, 444–453 (PMLR, 2019).
40. Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8024–8035 (Curran Associates, Inc., 2019).
41. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
42. Peng, H., Long, F. & Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* **27**, 1226–1238 (2005).
43. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018).

44. Mahendran, A. & Vedaldi, A. Salient deconvolutional networks. In *European Conference on Computer Vision*, 120–135 (Springer, 2016).
45. Molnar, C. *Interpretable machine learning* (Lulu. com, 2020).
46. Castro, J., Gómez, D. & Tejada, J. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research* **36**, 1726–1730 (2009).
47. Chollet, F. *et al.* Keras (2015).
48. Candès, E., Fan, Y., Janson, L. & Lv, J. Panning for gold: 'model-x' knockoffs for high dimensional controlled variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **80**, https://doi.org/10.1111/rssb.12265.
49. Pedregosa, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).
50. Kraskov, A., Stögbauer, H. & Grassberger, P. *Estimating mutual information. Physical review E* **69**, 066138 (2004).
51. Ding, C. & Peng, H. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology* **3**, 185–205 (2005).
52. Boonyakitanont, P., Lek-Uthai, A., Chomtho, K. & Songsiri, J. A review of feature extraction and performance evaluation in epileptic seizure detection using eeg. *Biomedical Signal Processing and Control* **57**, 101702 (2020).
53. Raimondi, D., Orlando, G., Vranken, W. F. & Moreau, Y. Exploring the limitations of biophysical propensity scales coupled with machine learning for protein sequence analysis. *Scientific reports* **9**, 1–11 (2019).
54. Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. *Classification and regression trees* (Routledge, 2017).
55. Domingos, P. A few useful things to know about machine learning. *Communications of the ACM* **55**, 78–87 (2012).
56. Vapnik, V. N. & Chervonenkis, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, 11–30 (Springer, 2015).
57. Bartlett, P. L., Harvey, N., Liaw, C. & Mehrabian, A. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research* **20**, 2285–2301 (2019).
58. Hua, J., Xiong, Z., Lowey, J., Suh, E. & Dougherty, E. R. Optimal number of features as a function of sample size for various classification rules. *Bioinformatics* **21**, 1509–1515 (2005).
59. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2623–2631 (2019).
60. Foret, P., Kleiner, A., Mobahi, H. & Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412 (2020).
61. Kwon, J., Kim, J., Park, H. & Choi, I. K. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, 5905–5914 (PMLR, 2021).
62. Wolpert, D. H. The supervised learning no-free-lunch theorems. *Soft computing and industry: Recent applications* 25–42 (2002).
63. Verplaetse, N., Passemiers, A., Arany, A., Moreau, Y. & Raimondi, D. Large sample size and nonlinear sparse models outline epistatic effects in inflammatory bowel disease. *Genome Biology* **24**, https://doi.org/10.1186/s13059-023-03064-y (2023).
64. Weiss, K., Khoshgoftaar, T. M. & Wang, D. A survey of transfer learning. *Journal of Big data* **3**, 1–40 (2016).

Author contributions

P.F., G.B., D.R. conceived the experiments, A.P., P.F. and D.R. conducted the experiments, A.P. and P.F. analyzed the results. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-82583-5.

Correspondence and requests for materials should be addressed to A.P., D.R. or P.F.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.