ARTICLES
https://doi.org/10.1038/s42256-021-00308-z

nature machine intelligence



# Variable selection with false discovery rate control in deep neural networks

Zixuan Song and Jun Li ✉

**Deep neural networks are famous for their high prediction accuracy, but they are also known for their black-box nature and poor interpretability. We consider the problem of variable selection, that is, selecting the input variables that have significant predictive power on the output, in deep neural networks. Most existing variable selection methods for neural networks are only applicable to shallow networks or are computationally infeasible on large datasets; moreover, they lack a control on the quality of selected variables. Here we propose a backward elimination procedure called SurvNet, which is based on a new measure of variable importance that applies to a wide variety of networks. More importantly, SurvNet is able to estimate and control the false discovery rate of selected variables empirically. Further, SurvNet adaptively determines how many variables to eliminate at each step in order to maximize the selection efficiency. The validity and efficiency of SurvNet are shown on various simulated and real datasets, and its performance is compared with other methods. Especially, a systematic comparison with knockoff-based methods shows that although they have more rigorous false discovery rate control on data with strong variable correlation, SurvNet usually has higher power.**

Deep neural networks (DNNs) are a popular machine-learning technique and have shown superior performance in many scientific problems. Despite their high prediction accuracy, DNNs are often criticized for a lack of interpretation of how changes of the input variables influence the output. Indeed, for applications in many scientific fields such as biology and medicine, understanding the statistical models described by the networks can be as important as, if not more important than, the prediction accuracy. With a DNN, because of its nonlinearity and inherent complexity, one should not expect a concise relationship between each input variable and the output, such as the conditional monotonicity in linear regression or logistic regression. A more realistic approach for interpreting the DNN model can be selecting a subset of variables, among all input variables, that have significant predictive power on the output, which is known as 'variable selection'. This paper considers the variable selection problem in DNNs.

The variable selection methods for neural networks (including but not limited to DNNs), similar to the ones for other machine learning techniques, can be broadly classified into three categories: filters, wrappers and embedded methods[1–3]. Filters select variables by information theoretic criteria, such as mutual information[4] and partial mutual information[5], and the selection procedure does not involve network training. By contrast, both wrappers and embedded methods are based on the training of neural networks. Wrappers wrap the training phase with a search strategy, which searches through the set, or a subset, of all possible combinations of input variables and selects the combination whose corresponding network gives the highest prediction accuracy. A number of sequential[6] and heuristic search strategies[7–9] have been used. Embedded methods, unlike wrappers, select variables during the training of the network of interest. This can be done by gradually removing/pruning weights or variables according to their importance measured in various ways (a detailed review is given in the Methods section) or by incorporating a regularization term into the loss function of the neural network to impose sparsity on the weights[10–13]. For a more exhaustive review of variable selection methods in neural networks, see refs. [1,14].

While a lot of variable selection methods have been developed for neural networks, there are still challenges that hinder them from being widely used. First and foremost, most of these methods lack a control on the quality of selected variables. When selecting from a large number of variables, a standard way of quality control is to calculate false discovery rate (FDR)[15] and control it at a certain level, particularly in biological and medical studies. In the context of variable selection, FDR is the (expected) proportion of false positives among all variables called significant; for example, if 20 variables are selected (called significant), and two of them are actually null, then the FDR is $2/20 = 0.1$. Currently, most methods do not provide FDR control, but there are notable exceptions: a few methods[16,17] utilize a modern FDR control framework based on 'knockoffs'[18,19] for controlled variable selection in neural networks. We will later study their performance and compare it with that of our method. Second, among these methods, many were developed for specific types of networks, especially very shallow networks, and they do not work, or work inefficiently, for deeper networks. Third, many of the methods are not applicable to large datasets, on which their computational loads can be prohibitively high.

In this paper, we develop a method called SurvNet for variable selection in neural networks that overcomes these limitations. It is an embedded method that gradually removes least relevant variables until the FDR of remaining variables reaches a desired threshold. Figure 1 shows the flowchart of SurvNet. It starts by adding a set of simulated input variables called 'surrogate variables' that help estimate the FDR and train a network with all variables, including both original and surrogate variables. Then it calculates the importance of each variable (original or surrogate) and eliminates the variables that are least important. When eliminating a variable, its corresponding input neuron and all outgoing connections of this neuron are removed from the network. After this, SurvNet estimates the FDR of the original variables that remain in the model. If the estimated FDR is greater than the pre-set threshold, SurvNet will go back to the step of training the (updated) network; otherwise, the elimination stops, and all remaining surrogate variables are

Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN, USA. ✉e-mail: jun.li@nd.edu

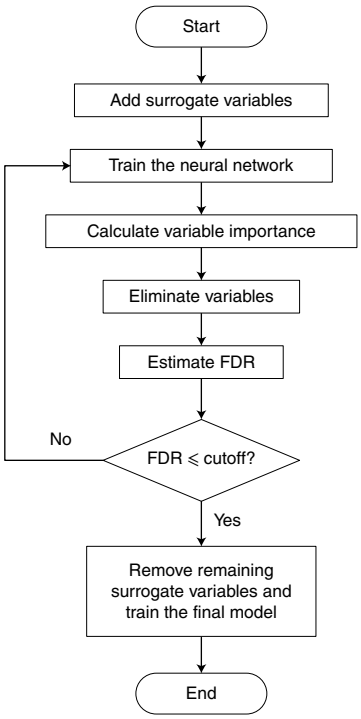


**Fig. 1 | Flowchart of SurvNet.** The variable selection procedure of SurvNet starts by adding surrogate variables. Then it calculates the importance of all variables in a trained network and eliminates a number of least important variables. Next, SurvNet estimates the FDR of remaining original variables. If the estimated FDR is greater than the pre-set cutoff, the elimination proceeds; otherwise, the selection procedure ends.

removed before the final model is trained. Note that each updated network is trained using the values of weights in the last trained network as initial values for a 'warm start'.

There are three major novelties in this backward elimination procedure of SurvNet. First, it proposes a new measure/score of variable importance, which works regardless of the type of problems (classification or regression), the number of output neurons (one or multiple), and the number of hidden layers (one or multiple) in neural networks. In fact, this score can be readily computed for networks with arbitrary depths and activation functions. Second, SurvNet proposes an easy and quick way of estimating FDRs. Statistical estimation of FDRs requires obtaining the null distribution of the importance scores, that is, the distribution of the scores of irrelevant variables[20]. This is often done by permuting the output values of samples and training multiple independent models in parallel, each of which corresponding to a permuted dataset, but the computational cost is typically unaffordable for neural networks. SurvNet proposes a distinct way: it generates a set of null variables, typically by random sampling from the original data matrix with or without replacement, which serve as surrogates of the (unknown) null original variables to obtain the null distribution. The idea of surrogate variables is similar to that of knockoffs, but they differ in several important aspects (discussed in detail in Supplementary Information). With the introduction of surrogate variables, an estimate of FDR can be given by a simple mathematical formula without training a large number of networks at each step. Third, at each step, instead of eliminating one variable or any pre-specified number of variables, SurvNet is able to adaptively determine an appropriate number of variables to eliminate. This number, expressed in a concise mathematical formula, makes the elimination highly efficient while having the estimated FDR well controlled on the desired level. The formula includes a parameter called 'elimination rate', which is a constant between 0 and 1 and controls the 'aggressiveness' of elimination. When this parameter is chosen to be 1, the elimination is the most aggressive, and the number of steps needed to reach the desired FDR level is expected to be the least.

Put together, SurvNet is a computationally efficient mechanism for variable selection in neural networks that needs little manual intervention. After setting the initial network structure, an FDR cutoff $\eta^*$ (0.1 is the most commonly used value), and an elimination rate $\varepsilon$ (1 is often an acceptable choice), the elimination procedure will automatically determine how many and which variables to eliminate at each step and stop when the estimated FDR is no greater than $\eta^*$.

## Performance of SurvNet on simulated data

We first applied SurvNet to four datasets simulated under different schemes (datasets 1–4). Datasets 1–3 were for classification and dataset 4 was for regression. For dataset 1, we simulated a $10{,}000 \times 784$ matrix, each element of which followed a uniform distribution on (0,1), and treated its rows and columns as samples and variables respectively. The samples were randomly assigned into two classes of equal size, and $p' = 64$ variables were chosen at random with their values in one class being shifted by a small amount between 0.1 and 0.3. In this way, the 784 variables were independent from each other, and the 64 chosen variables were significant because each of them had different mean values in the two classes. This 'independent-variable differential-mean' scheme is widely used for studying variable selection. For dataset 2, we considered correlated variables, since it is well known that variable dependence often makes FDR estimation difficult[21,22]. As the pixel value of image data usually highly depends on the values of surrounding pixels, here we used all images of digit 0 in the MNIST data[23] and randomly assigned them into two classes. Then we picked $p' = 64$ variables and shifted their mean values in one class. Dataset 3 was very challenging: unlike in the previous two datasets, the significant variables did not differ in the mean values of two classes; instead, they differed only in the variances. In other words, the only difference between the two classes was that 64 out of 784 variables were made to be 'noisier' with their standard deviations being inflated from 0.29 to 0.95 (see Supplementary Information for calculations). In this case, classifiers and tests that detect discrepancies in the mean values, such as the $t$-test, would fail. Dataset 4 was a regression dataset. It was a $10{,}000 \times 784$ matrix whose elements were uniformly distributed on $(-1,1)$, and 64 of the 784 variables were randomly chosen as significant variables. The response variable depended on the main effects and interactions of the significant variables as well as their nonlinear transformations. See Methods for details of the simulation schemes.

On these simulated data, the performance of SurvNet was evaluated by the number of significant variables selected, the estimated and actual FDR of selected variables, as well as the initial and final test error, which were the misclassification rate or the mean squared error using the network with all original variables and with the selected variables only, respectively. See Supplementary Information for details about how they were calculated.

To demonstrate how our method works step by step, we ran SurvNet on dataset 1 with an FDR cutoff $\eta^* = 0.1$ and an elimination rate $\varepsilon = 1$, and Fig. 2a shows, in one instance of simulation, the number of original variables and surrogate variables left at each step of the selection process as well as the corresponding estimated FDR. Also displayed in the figure is the number of variables to be eliminated in the subsequent step, which indicates that our algorithm was efficient: it eliminated a large number of variables at the beginning and gradually slowed down the elimination as the number of remaining variables decreased and the estimated FDR got closer to the desired value. When the estimated FDR became less than 0.1,

**a**

| Step | $r-r_0$ | $r_0$ | $\hat{\eta}$ | $m$ | $r'$ |
|---|---|---|---|---|---|
| 0 | 784 | 784 | 1.000 | 706 | 64 |
| 1 | 441 | 421 | 0.955 | 377 | 64 |
| 2 | 257 | 228 | 0.887 | 203 | 64 |
| 3 | 165 | 117 | 0.709 | 101 | 64 |
| 4 | 113 | 68 | 0.602 | 57 | 64 |
| 5 | 86 | 38 | 0.442 | 30 | 64 |
| 6 | 77 | 17 | 0.221 | 10 | 64 |
| 7 | 72 | 12 | 0.167 | 5 | 64 |
| 8 | 69 | 10 | 0.145 | 4 | 64 |
| 9 | 68 | 7 | 0.103 | 1 | 64 |
| 10 | 68 | 6 | 0.088 | Stop | 64 |

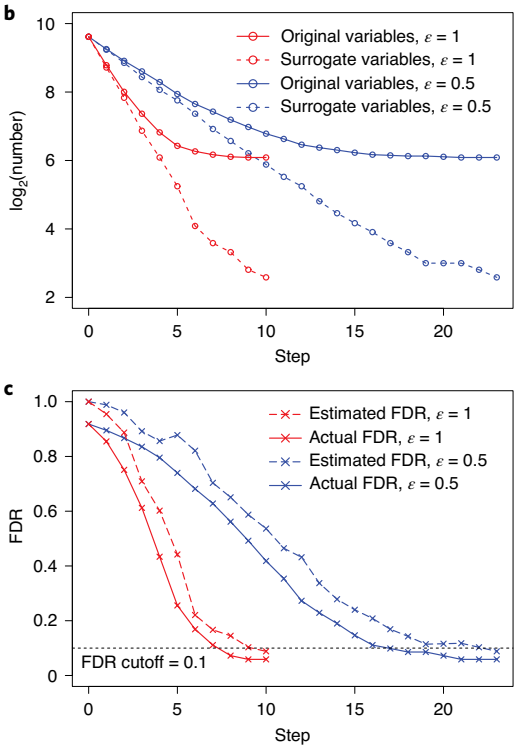


**Fig. 2 | Variable selection process on dataset 1. a**, The number of original variables ($r-r_0$), surrogate variables ($r_0$), and significant variables ($r'$) left at each step of the selection process, together with the estimated FDR ($\hat{\eta}$) and the number of variables to be eliminated in the next step ($m$), when $p' = 64$, $\eta^* = 0.1$, and $\varepsilon = 1$. **b**, The number of original and surrogate variables along the selection processes with different elimination rates when $p' = 64$ and $\eta^* = 0.1$. **c**, The estimated and actual value of FDR along the selection processes with different elimination rates when $p' = 64$ and $\eta^* = 0.1$.

the selection process stopped, and the final model turned out to contain all 64 truly significant variables. On the same data, we studied the influence of elimination rates, and the results of using $\varepsilon = 1$ and $\varepsilon = 0.5$ are shown in Fig. 2b,c. It is found that while a larger elimination rate led to a faster selection process with fewer steps, the number of variables left at the end of the selection was almost the same (Fig. 2b). Moreover, regardless of the elimination rate, our method gave an accurate estimate of FDR, and the true value of FDR was well controlled throughout the selection process (Fig. 2c).

The overall performance of SurvNet under $\eta^* = 0.1$ and $\varepsilon = 1$ on different datasets was summarized in Table 1. SurvNet accurately selected the significant variables: the FDR of selected variables was always close to the cutoff value 0.1, and the estimated FDR was also accurate, that is, close to the actual FDR. Except for dataset 3, SurvNet always picked out more than 90% of the significant variables. In dataset 3, although SurvNet only successfully identified $23/64 \approx 36\%$ of the significant variables, it was still much superior to the $t$-test, which merely identified 0.20 (averaged over 25 simulations) significant variables, that is, $0.20/64 \approx 0.31\%$ of all significant variables.

We scrutinized the selection process of SurvNet on dataset 3 and found the reason why only a proportion of significant variables were retained: the initial network, which made almost random guesses on the output, could not accurately determine the importance of input variables to the output, and thus many significant variables were removed at the first elimination step. As the selection proceeded, the network gained higher classification accuracy and also stronger ability to distinguish the significant variables; as a result, the false elimination of significant variables became less likely. Given this reason, SurvNet should be able to keep a larger proportion of significant variables if a smaller elimination rate, say $\varepsilon = 0.5$, was used. We found that this was indeed the case (see Supplementary Table 3 for details).

On all the simulation datasets, the prediction accuracy of the network was improved after variable selection. In particular, there was a dramatic improvement of classification accuracy on dataset 3 (data with variance-inflated variables): while the test error given by the network with all 784 variables was 49.42%, it dropped to 0.47% after variable selection by SurvNet; that is, from an almost random guess to an almost perfect classification. This implies that the variable selection gave back to the DNN the ability to utilize all types of information useful for classification, which was masked by the overwhelming irrelevant variables.

The results under different elimination rates ($\varepsilon = 1$ and $\varepsilon = 0.5$), different FDR cutoffs ($\eta^* = 0.1$ and $\eta^* = 0.05$), and different numbers of significant variables ($p' = 64$ and $p' = 32$) on datasets 1–4 are shown in Supplementary Tables 1–4, respectively.

## Performance of SurvNet on real data

After four simulation datasets, we then applied SurvNet to digits 4 and 9 in the MNIST database (dataset 5) and a single-cell RNA-Seq dataset (dataset 6).

MNIST contains 70,000 images of ten handwritten digits from 0 to 9, each of which contains $28 \times 28 = 784$ pixels, which are treated as 784 input variables. Here we only used the images of two digits that look alike (4 and 9), as they are similar in most pixels and are only different in pixels in certain regions. In the top panel of Fig. 3a, we show two representative 9s that differ in the presence of a bottom hook and two representative 4s that differ in the width of top opening. The four regions circled in red are likely to be most significant in differentiating 4s and 9s, especially the region in the upper middle denoting whether the top is closed or open, and the region in the lower middle denoting whether there is a hook at the bottom.

From left to right, the bottom panel of Fig. 3a shows the pixels that were selected by SurvNet under four combinations of FDR

**Table 1 | Summary statistics of results of SurvNet on datasets 1–6**

| | Test error | | Selected variables (no.) | | FDR | |
|---|---|---|---|---|---|---|
| | Initial | Final | Total | Significant | Estimated | Actual |
| Dataset 1 | 0.36% | 0.27% | 69.36 | 61.92 | 0.093 | 0.105 |
| | (0.17%) | (0.10%) | (5.07) | (2.48) | (0.004) | (0.044) |
| Dataset 2 | 0% | 0% | 66.88 | 59.36 | 0.094 | 0.107 |
| | (0%) | (0%) | (8.73) | (5.87) | (0.005) | (0.057) |
| Dataset 3 | 49.42% | 0.47% | 26.40 | 23.00 | 0.076 | 0.114 |
| | (1.69%) | (0.48%) | (13.68) | (11.87) | (0.031) | (0.089) |
| Dataset 4[a] | 33.013 | 8.901 | 71.16 | 63.96 | 0.094 | 0.097 |
| | (27.059) | (1.988) | (5.02) | (0.20) | (0.004) | (0.061) |
| Dataset 5[b] | 1.69% | 1.71% | 69.52 | - | 0 | - |
| | (0.24%) | (0.21%) | (13.35) | - | (0.002) | - |
| Dataset 6[b] | 0.083% | 0.076% | 149.44 | - | 0.007 | - |
| | (0.074%) | (0.082%) | (49.62) | - | (0.003) | - |

The numbers are averaged over 25 simulations/runs, with corresponding standard deviations in parentheses. $p' = 64$, $\eta^* = 0.1$ and $\varepsilon = 1$ are used for the simulated datasets (datasets 1–4); $\eta^* = 0.01$ and $\varepsilon = 1$ are used for the real datasets (datasets 5 and 6). [a]Dataset 4 is a regression dataset and thus the test error is measured by the mean squared error. [b]Datasets 5 and 6 are real datasets, and thus the number of significant variables and the actual FDR are unknown.

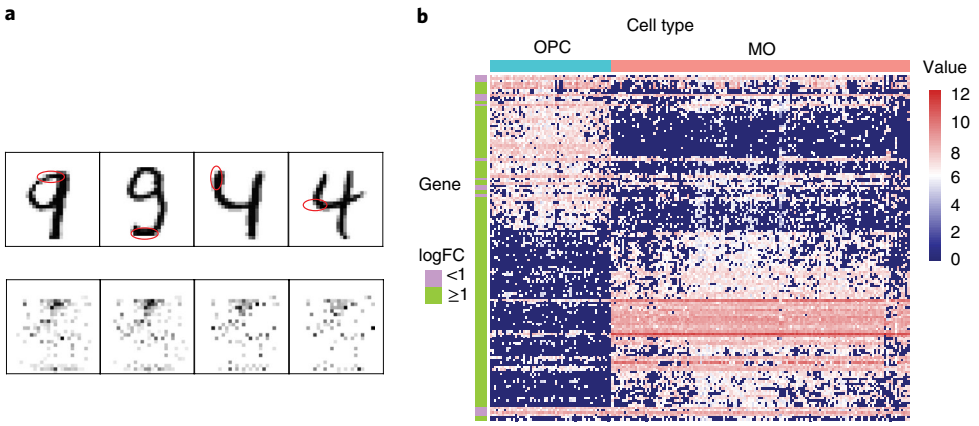


**Fig. 3 | Results of SurvNet on the two real datasets. a**, The top panel shows examples of hand-written digits 4 and 9 (two images for each). The circles mark the locations of distinctive pixels of these two digits. The bottom panel shows heatmaps of 28 × 28 pixels under four conditions with different FDR cutoffs and elimination rates, which display the relative importance of remaining pixels. The darker the colour of a pixel, the more important it is. The corresponding conditions are (from left to right): $\eta^* = 0.1$, $\varepsilon = 1$; $\eta^* = 0.1$, $\varepsilon = 0.5$; $\eta^* = 0.01$, $\varepsilon = 1$; $\eta^* = 0.01$, $\varepsilon = 0.5$. **b**, Heatmap showing the expression of selected genes in the single-cell RNA-Seq dataset in two groups of cells. Rows represent individual genes and columns are 200 randomly chosen cells. The genes whose log fold changes in OPCs and MOs are less than 1 are distinguished from others.

cutoffs ($\eta^* = 0.1$ or 0.01) and elimination rates ($\varepsilon = 1$ or 0.5). The colours display the relative importance, defined by equation (2) (see Methods), of the selected pixels, and a darker colour means greater importance. We found that different parameter settings gave quite consistent results, and they all picked out the four regions that were speculated to be significant.

Single-cell RNA-Seq[24] is a biological technique for measuring gene expression in cells. In single-cell RNA-Seq data, the samples are the cells, the inputs are the expression levels of individual genes, and the output is the cell type. Chen et al. performed single-cell RNA-Seq analysis of the adult mouse hypothalamus and identified 45 cell types based on clustering analysis[25]. For dataset 6, we used 5,282 cells in two non-neuronal clusters, oligodendrocyte precursor cell (OPC) and myelinating oligodendrocyte (MO), which reflected two distinct stages of oligodendrocyte maturation. After preprocessing (described in Supplementary Information), 1,046 genes were left for further analysis.

With $\eta^* = 0.01$ and $\varepsilon = 1$, SurvNet selected 145 genes in one realization. Figure 3b shows a heatmap of the expression values of these genes, in which rows are genes and columns are cells. The top banner shows the true class labels of the samples. For gene expression data, the set of significant genes is typically identified by 'differential expression' analysis, which finds differences in the mean expression levels of genes between classes. Indeed, as the heatmap shows, most genes have evidently different mean expression levels in the OPCs and MOs. However, among the 145 significant genes identified by SurvNet, 16 had log-fold-changes (logFCs) less than 1, meaning that their average expression values were not very different in the two classes. In Fig. 3b, these genes are marked in purple on the left banner, in contrast to green for the other genes. In fact, Bartlett's test, which tests the difference in variance, claimed that 14 of these 16 genes had unequal variances in the two groups of cells ($p < 0.05$); thus, they were actually instances of variance-inflated variables selected by SurvNet, in addition to the ones in dataset 3. Again,

**Table 2 | Performance of SurvNet, GL and SGL on datasets 1–6**

| | Total selected variables (no.) | | | Final test error | | | Actual FDR | | |
|---|---|---|---|---|---|---|---|---|---|
| | SurvNet | GL | SGL | SurvNet | GL | SGL | SurvNet | GL | SGL |
| Dataset 1 | 69.36 | 76.88 | 60.80 | 0.27% | 8.80% | 22.52% | 0.105 | 0.388 | 0.399 |
| Dataset 2 | 66.88 | 23.36 | 73.92 | 0% | 0% | 0% | 0.107 | 0.175 | 0.501 |
| Dataset 3 | 26.40 | 8.80 | 20.44 | 0.47% | 50.72% | 50.49% | 0.114 | 0.899 | 0.900 |
| Dataset 4[a] | 71.16 | 58.40 | 75.88 | 8.90 | 83.59 | 78.66 | 0.097 | 0.887 | 0.714 |
| Dataset 5[b] | 69.52 | 130.76 | 62.40 | 1.71% | 17.45% | 34.65% | - | - | - |
| Dataset 6[b] | 149.44 | 143.28 | 127.36 | 0.076% | 0.085% | 0.112% | - | - | - |

The numbers are averaged over 25 simulations/runs. For GL and SGL, since they cannot estimate FDR, proper regularization strength is given in each experiment so that roughly the same number of variables are removed when SurvNet is applied (with $\eta^* = 0.1$ for datasets 1–4, $\eta^* = 0.01$ for datasets 5 and 6). All simulated datasets (datasets 1–4) have 64 significant variables. [a]Dataset 4 is a regression dataset, and thus the test error is measured by the mean squared error. [b]Datasets 5 and 6 are real datasets, and thus the actual FDR is unknown.

SurvNet demonstrates its ability to identify various types of significant variables, not just variables with different means. Further, the functional interpretations of the selected genes match the biological characteristics of OPCs and MOs (see Supplementary Information).

## Comparisons with regularization and knockoff-based methods

We first compared SurvNet with two state-of-the-art embedded methods: group lasso and sparse group lasso regularizations for DNNs[13]. In these two methods, which we call GL and SGL for short, weights from each neuron are grouped, and the grouped weights are regularized towards being simultaneously zero, so that some entire neurons (including input neurons) can be removed from the network.

GL and SGL do not estimate and control FDR, and thus they cannot determine the number of variables to select. This is a crucial disadvantage in practice. This inability to determine a proper regularization strength ($\lambda$) also makes their comparison with SurvNet non-trivial: for the comparison, which $\lambda$ should we use for GL and SGL? We circumvented this difficulty in two ways. First, we manually set $\lambda$ so that GL and SGL kept a set of variables of similar size as SurvNet, that is, we 'lent' them the ability of SurvNet to determine a proper number of variables to select, and checked whether their selected variables were truly significant and whether they predicted the outcome accurately. Table 2 gives the results of the three methods. It is clear that SurvNet performed the best on all the six datasets. Note that the total number of selected variables, which was determined by the $\lambda$ value we set, is not a valid criterion for the performance. On most datasets, the gaps between the performance of SurvNet and the other two methods were quite large. This means that the performances of GL and SGL were much inferior to SurvNet, even when we equipped them with the ability to determine the number of variables to select. Is it possible that they performed better under another $\lambda$ value? To study this, in our second way, we tried a series of $\lambda$ values for GL and SGL. We found, interestingly, that a low FDR had never been achieved, no matter what $\lambda$ value was used. The likely reason is discussed in Supplementary Information.

Next, we compared SurvNet with variable selection methods based on knockoffs, which are a hot research topic in the field of FDR control. Similar to surrogate variables, knockoffs serve as negative controls for the original variables, but they are constructed to further preserve the correlations between the original variables, and thus it is guaranteed, theoretically, that the FDR is controlled under arbitrary variable correlation. Knockoff-based methods typically select variables in a single run, that is, by a one-step procedure without re-training of networks. Here, we first compared SurvNet with two naive ways of combining SurvNet with knockoff samples, which were generated using two representative approaches called 'second-order' knockoffs[19] and 'deep' knockoffs[26], then we compared SurvNet with two representative methods that apply knockoffs in neural networks: one[16] designed a new DNN architecture called DeepPINK, and the other[17] developed an efficient algorithm to sample valid knockoffs for Bayesian models as well as new knockoff test statistics. The comparisons were done on all our simulation datasets and the datasets from[16,17]. See Supplementary Information for more details. Below, the performance is summarized in two most important aspects: FDR control and power.

Regarding FDR control, both SurvNet and knockoff-based methods successfully controlled FDR in most but not all cases. Knockoff-based methods failed on dataset 3 in our paper, that is, the data with variance-inflated variables, where the significant variables were not only sparse but also 'weak', in the sense that they did not have strong and easy-to-capture effects on the outcome. Knockoff-based methods' failure on such challenging data was likely due to the one-step procedure they used to select variables, and SurvNet overcame this difficulty by its (multi-step) backward elimination procedure. SurvNet failed on a synthetic dataset in ref. [17] where the correlations between the variables were exceedingly strong. This was likely due to the use of permutations, which sacrificed the correlations between variables, to generate surrogate variables in the current version of SurvNet. While its failure on the synthetic dataset in ref. [17] implies that a theoretical guarantee of SurvNet to control the FDR under arbitrary variable correlation may not exist, its control of FDR was flawless on all the other datasets where the variables were known to be correlated, including our dataset 2 (MNIST data) and the synthetic data in ref. [16], as well as the HIV-1 drug resistance data in ref. [16], which was the only real dataset where the ground truth was (considered as) known.

As to power, SurvNet usually has higher power than knockoff-based methods. This was observed in the comparisons on all the simulated datasets and the only real dataset where the true answer was (considered as) known. For the other real datasets, where the true answers were unknown, SurvNet reported a larger number of selected variables. In many cases, the improvement in power was substantial.

A systematic description and discussion of our comparisons is given in Supplementary Information.

## Conclusions and discussion

We have presented a largely automatic procedure for variable selection in neural networks (SurvNet). It is based on a new measure of variable importance that applies to a variety of networks, deep or shallow, for regression or classification, and with one or multiple output units. More importantly, SurvNet aims to estimate and control the FDR of selected variables in neural networks, which is essential for applications where the trustworthiness of variable selection is pivotal. By introducing surrogate variables, it avoids training multiple networks in parallel. SurvNet also adjusts the number of

variables to eliminate at each step, and the 'warm start' nature of backward elimination facilitates the training of networks. On multiple simulation datasets and real datasets, SurvNet has effectively identified the significant variables. It has given a dependable estimate of FDR as well, in almost all datasets we considered.

SurvNet takes advantages of modern developments of DNNs. The importance scores of input variables that are based on derivatives with respect to the inputs can be efficiently computed by functions in deep-learning packages such as TensorFlow, PyTorch, and Theano. Moreover, advances in optimization techniques and computation platforms have made the training of DNNs highly scalable. In particular, DNNs can accommodate a large number of input variables, which enables the introduction of surrogate variables.

## Methods

**Measures of variable importance.** *Notation.* We use a tuple $(\boldsymbol{x},\boldsymbol{y})$ to represent the input and the output of the network, with $\boldsymbol{y}$ being either one-dimensional or multi-dimensional. $x_j$ denotes the $j$th component of $\boldsymbol{x}$, namely the $j$th variable, and $(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)})$ $(i=1,\ldots,n)$ is the $i$th sample, where $n$ is the total number of samples (in the training set). Given a proper form of the loss $L(\cdot,\cdot)$, the loss function $L^* = \sum_{i=1}^{n} L(\boldsymbol{y}^{(i)}, f(\boldsymbol{x}^{(i)}))$, where $f$ denotes the output function of the network. The most popular choices for $L(\cdot,\cdot)$ are the squared error loss for regression problems and the cross-entropy loss for classification problems.

*Existing measures.* Many statistics have been proposed to measure the importance of variables in neural networks, and they generally fall into two categories[27,28].

One category of methods estimate the importance of $x_j$, denoted by $S_j$, based on the magnitudes of the connection weights in the network[29–33]. A simple example is the sum of absolute values of input weights[29], but larger values of weights in the input layer do not mean greater importance if connections in hidden layers have small weights, and a better alternative is to replace the input weights with the products of the weights on each path from this input to the output[30]. These measures were developed for networks with only one hidden layer, and they are unlikely to work well for deeper networks as the outgoing weights of a neuron does not reflect its importance once the neuron is inactive (for example, when the input of a sigmoid neuron is far from zero or the input of a ReLU neuron is negative).

The other category of methods estimate $S_j$ by the sum of influences of the input weights on the loss function, that is $S_j = \sum_{k\in\Omega_j} \delta L_k^*$, where $\boldsymbol{\Omega}_j$ is the set of outgoing weights from the $j$th input neuron, and $\delta L_k^*$ is the increment of the loss function caused by the removal of weight $w_k$ (ref. [27]). $\delta L_k^*$ can be approximated by a Taylor series of the loss function using first-order[34,35] or second-order terms[36–38]. However, it is unclear why $S_j$ equals the (unweighted) sum of $\delta L_k^*$.

Apart from these two major categories of measures, it was also proposed to use $S_j = \frac{\partial f}{\partial x_j}$, that is $S_j = \frac{\partial y}{\partial x_j}$, when the output $y$ is one-dimensional[39,40]. But it is unclear how $S_j$ should be defined when there are multiple output units. Let $y_1,\ldots,y_K$ be the output values of $K$ output units, and one definition of $S_j$ was given by $S_j = \sum_{k=1}^{K} |\frac{\partial y_k}{\partial x_j}|$ (ref. [41]). However, using this summation seems problematic in some cases, especially when $y_1,\ldots,y_K$ are the outputs of softmax functions.

*Our new measure.* We propose a simple and direct measure of the importance of variable $j$ based on $\frac{\partial L}{\partial x_j}$, which describes how the loss changes with $x_j$. There are a few advantages of using $\frac{\partial L}{\partial x_j}$. First, regardless of the structure of the network and whether the output(s) is continuous or categorical, $L$ is always well defined since it is the target for the optimization/training of the network. Thus the proposed measure is applicable to a wide variety of networks. Second, no matter how many output units there are, $L$ is always a scalar and hence $\frac{\partial L}{\partial x_j}$ is always a scalar. There is no difficulty in combining effects from multiple output units. Third, $\frac{\partial L}{\partial x_j}$ is easily computable with backpropogation, and popular frameworks/libraries for DNN computations (for example, TensorFlow, PyTorch and Theano) all use differentiators that efficiently compute partial derivatives (gradients) of arbitrary forms.

Note that $\frac{\partial L}{\partial x_j}$ is a function of the tuple $(\boldsymbol{x},\boldsymbol{y})$, and hence it is natural to estimate it by its mean over all observations in the training set. To avoid cancellation of positive and negative values, we measure the importance of $x_j$ by the mean of absolute values

$$S_j = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{\partial L}{\partial x_j}(\boldsymbol{y}^{(i)}, f(\boldsymbol{x}^{(i)}))\right|, \tag{1}$$

or the mean of squares

$$S_j = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{\partial L}{\partial x_j}(\boldsymbol{y}^{(i)}, f(\boldsymbol{x}^{(i)}))\right)^2, \tag{2}$$

where $\frac{\partial L}{\partial x_j}(\boldsymbol{y}^{(i)}, f(\boldsymbol{x}^{(i)}))$ is the value of $\frac{\partial L}{\partial x_j}$ at the $i$th training sample.

The importance scores given by equations (1) and (2) implicitly assume that all the input values have similar range, which is typically the case for DNNs, since it is common practice to standardize/scale the variables before supplying them to the network for the sake of faster and more stable training of the network[42,43]. If this is not the case, we suggest the score in equation (1) be multiplied by the (sample) standard deviation of $x_j$ and the score in equation (2) be multiplied by the (sample) variance of $x_j$.

Note that in the case of multiple linear regression, $L = \frac{1}{2}(y-\hat{y})^2 = \frac{1}{2}(y-\sum_j \beta_j x_j)^2$, where $y$ is a scalar response and $\beta_j$ is the $j$th regression coefficient, then $\frac{\partial L}{\partial x_j} = -(y-\hat{y})\beta_j$. Thus, $S_j$ is defined as $|\beta_j| \times \frac{1}{n}\sum_{i=1}^{n}|e_i|$ or ${\beta_j}^2 \times \frac{1}{n}\sum_{i=1}^{n} {e_i}^2$ by (1) and (2) respectively, where $e_i = y^{(i)} - \hat{y}^{(i)}$. Note that $S_j$ is proportional to $|\beta_j|$ or ${\beta_j}^2$ as $\frac{1}{n}\sum_{i=1}^{n}|e_i|$ and $\frac{1}{n}\sum_{i=1}^{n} {e_i}^2$ are constants. Therefore, both of them are reasonable measures of the contribution of the $j$th variable, and they are actually equivalent in this case. The meaning of $S_j$ in some other special cases, such as linear regression with multiple outputs and logistic regression with one or multiple outputs, is elaborated in Supplementary Information.

All results in the main text were obtained using equation (2). Results obtained using equation (1) (Supplementary Information) are not notably different.

**Elimination procedure with FDR control.** In this section, we first introduce how we estimate FDR and then talk about how we use this estimate to determine the number of variables to eliminate at each step.

*Introduction of surrogate variables.* The key of estimating FDR[20] is to estimate/generate the null distribution of the test statistic. In our case, it is to obtain the distribution of the importance score $S_j$ defined by equation (2) or equation (1) for variables that are not significant. Since the network is a complicated and highly nonlinear model, a theoretical distribution that applies to various network structure and various types of data may not exist. This null distribution needs to be obtained for the network and the data in hand.

However, it is usually unknown which variables are truly null. If we construct the null distribution by permuting the output values of the data, it seems inevitable to train multiple networks from scratch in parallel. For this reason, we propose to introduce/add a number of variables that are known/generated to be null. We call these variables 'surrogate null variables' (or 'surrogate variables' for short). These variables will be concatenated with the original variables to form a larger data matrix.

To be precise, suppose there are $p$ original variables and $n$ training samples (including validation samples). Then after we add $q$ surrogate variables, the new data matrix will be of size $n\times(p+q)$, which binds the original $n\times p$ data matrix $\boldsymbol{X}$ with an $n\times q$ data matrix for surrogate variables $\boldsymbol{X}_s$. It is assumed that the original variables are distributed in similar ranges or have been standardized, which is a suggested pre-processing step as it benefits the training of the network, and the elements in $\boldsymbol{X}_s$ are sampled with replacement (or without replacement when $q \le p$) from the elements in $\boldsymbol{X}$. As a result, the $q$ surrogate variables are null, and their importance scores give the null distribution.

We recommend $q$ to be on the same scale as $p$ (see Supplementary Information for a more detailed discussion about the choice of $q$). For convenience, $q$ takes the same value as $p$ in all experiments in this paper. In this case, the elements in $\boldsymbol{X}_s$ can be generated by permuting the elements in $\boldsymbol{X}$.

The selection procedure of SurvNet starts with using all $p+q$ variables as inputs. Then at each step, it eliminates a number of least important variables, including both original variables and surrogate variables. The remaining variables are used to continue training the network, and the elimination stops once the FDR falls below the cutoff.

*FDR estimation.* Then we consider how to estimate FDR at any given time of the selection process. Suppose $r$ variables are retained in the network, among which there are $r_0$ surrogate variables, then $r_0/q$ proportion of surrogate (null) variables have not been eliminated yet. Accordingly, one would expect that roughly the same proportion of null original variables still exist at this time, that is, approximately $\frac{r_0}{q} \times p_0$ variables among the remaining original variables are falsely called significant, where $p_0$ is the number of null variables in the original dataset. Thus, an estimate of the FDR of the $r-r_0$ original variables is given by

$$\tilde{\eta} = \frac{\frac{r_0}{q}\times p_0}{r-r_0}. \tag{3}$$

In practice, however, $p_0$ is unknown, and a common strategy is to replace it with its upper bound $p$ (ref. [20]). Hence we have the following estimated FDR,

$$\hat{\eta} = \frac{\frac{r_0}{q}\times p}{r-r_0} = \frac{r_0}{r-r_0}\times\frac{p}{q}, \tag{4}$$

which is greater than $\tilde{\eta}$, but the difference is negligible when the proportion of relevant variables is small. That is, $\frac{\hat{\eta}}{\tilde{\eta}} = \frac{p}{p_0} \approx 1$ when $\frac{p-p_0}{p}$ is close to zero. Apparently, when $\hat{\eta}$ is controlled to be no greater than a pre-specified threshold $\eta^*$, $\tilde{\eta}$ is guaranteed to be no greater than $\eta^*$ as well. When $q=p$, $\hat{\eta}$ can be simplified as $\frac{r_0}{r-r_0}$.

*Determination of the number of variables to eliminate.* If the estimated FDR $\hat{\eta}$ (given by equation (4)) is less than or equal to the FDR cutoff $\eta^*$, the variable

selection procedure stops. Otherwise, the procedure proceeds, and we want to decide how many variables to eliminate among the $r$ variables that are still in the model. Let this number be $m$, and the determination of $m$ is based on the following considerations. On one hand, we expect that the elimination process is time-saving and reaches the FDR threshold quickly; on the other hand, we want to avoid eliminating too many variables at each step, in which case the FDR may fall much lower than the threshold. We have,

*Claim 1. If $m$ variables are further eliminated from the current model, the smallest possible estimated FDR after this step of elimination is*

$$\min \hat{\eta}^{\text{new}} = (1 - \frac{m}{r_0}) \times \hat{\eta}, \tag{5}$$

*where $r_0$ is the number of surrogate variables that are in the model before this step of elimination.*

Proof. Suppose there are $m_0$ surrogate variables among the $m$ variables to be eliminated, $0 \le m_0 \le m$, then according to equation (4), $\hat{\eta}$ will be updated to

$$\hat{\eta}^{\text{new}} = \frac{r_0 - m_0}{r - r_0 - (m - m_0)} \times \frac{p}{q}. \tag{6}$$

Note that $\hat{\eta}^{\text{new}}$ is monotonically decreasing with respect to $m_0$ for any fixed $m$, we have

$$\min \hat{\eta}^{\text{new}} = \hat{\eta}^{\text{new}}|_{m_0=m} = \frac{r_0 - m}{r - r_0} \times \frac{p}{q}. \tag{7}$$

Equation (4) indicates that $\frac{1}{r-r_0} \times \frac{p}{q} = \frac{\hat{\eta}}{r_0}$. Plugging it into equation (7), we have

$$\min \hat{\eta}^{\text{new}} = (r_0 - m) \times \frac{\hat{\eta}}{r_0} = (1 - \frac{m}{r_0}) \times \hat{\eta}.$$

It follows from equation (5) that $\min \hat{\eta}^{\text{new}} = \eta^*$ when $m = (1 - \frac{\eta^*}{\hat{\eta}}) \times r_0$. Also, note that $\min \hat{\eta}^{\text{new}}$ is a monotonically decreasing function of $m$. Therefore, when $m < (1 - \frac{\eta^*}{\hat{\eta}}) \times r_0$, $\min \hat{\eta}^{\text{new}} > \eta^*$ and thus $\hat{\eta}^{\text{new}} > \eta^*$. That is,

*Corollary 1. When $m < \left(1 - \frac{\eta^*}{\hat{\eta}}\right) \times r_0$, the estimated FDR after this step of elimination $\hat{\eta}^{\text{new}}$ is guaranteed to be still greater than the FDR cutoff $\eta^*$.*

On the other hand, when $m \ge (1 - \frac{\eta^*}{\hat{\eta}}) \times r_0$, $\min \hat{\eta}^{\text{new}} \le \eta^*$. That is,

*Corollary 2. When $m \ge (1 - \frac{\eta^*}{\hat{\eta}}) \times r_0$, the estimated FDR after this step of elimination $\hat{\eta}^{\text{new}}$ may reach the FDR cutoff $\eta^*$.*

Corollary 1 says that $m$ being less than $(1 - \frac{\eta^*}{\hat{\eta}}) \times r_0$ is 'safe' but the elimination will not stop after this step. Corollary 2 says that $m$ being much larger than $(1 - \frac{\eta^*}{\hat{\eta}}) \times r_0$ may not be 'safe' anymore. Taking both into consideration, we choose the step size to be

$$m = \lceil (1 - \frac{\eta^*}{\hat{\eta}}) \times r_0 \rceil, \tag{8}$$

where $\lceil \cdot \rceil$ denotes 'ceiling', that is the smallest integer that is no less than $\cdot$. Notice that when $\hat{\eta} > \eta^*$, which is the premise of continuing to eliminate variables, $1 - \frac{\eta^*}{\hat{\eta}} > 0$, and $r_0 > 0$ as well since $\hat{\eta}$ is positive. Thus $m$ is ensured to be no less than 1 at each step of variable elimination.

This form of $m$ seems to be reasonable for the following reasons. First, if there remain a great number of surrogate variables in the network, clearly more of them should be taken out. As $r_0$ decreases, $m$ will be smaller, and this makes sense since one should be more careful in further elimination. Second, when $\hat{\eta}$ is much higher than $\eta^*$, one will naturally expect a larger $m$ so that the updated estimated FDR will approach this cutoff.

Using the $m$ determined by equation (8), there is a chance that the estimated FDR will get to the cutoff in only one step. Oftentimes such a fast pace is not preferred as removing too many inputs at a time may make our warm start of the training not warm any more. Hence we may introduce an 'elimination rate' $\varepsilon$, which is a constant between 0 and 1, and take

$$m = \lceil \varepsilon \times (1 - \frac{\eta^*}{\hat{\eta}}) \times r_0 \rceil. \tag{9}$$

**Experimental setup.** *Simulation schemes.* For dataset 1, we simulated a $10{,}000 \times 784$ matrix $\boldsymbol{X}$, with $x_{ij} \sim$ i.i.d. (independent and identically distributed) $U(0,1)$ for $1 \le i \le 10{,}000$, $1 \le j \le 784$, where $U$ means uniform distribution. The samples were randomly assigned into two classes $C_1$ and $C_2$ of equal size. Then $p' = 64$ variables were chosen at random and their values in one class were shifted: for each of these variables, we generated a shift value $\delta_j \sim U(0.1, 0.3)$, with its direction having equal probability of being positive and negative. More precisely, $x_{ij} \leftarrow x_{ij} + (2\alpha_j - 1) \times \delta_j$ for $i \in C_1$, $j \in \Omega_{p'}$, where $\alpha_j \sim$ Bernoulli($\frac{1}{2}$) and $\Omega_{p'}$ was the set of $p'$ randomly chosen variables.

For dataset 2, we used all images of digit 0 in the MNIST data and randomly assigned them into two classes. Then we picked $p' = 64$ variables and shifted their mean values in one class in the same way we did in dataset 1.

The third simulation scheme is very challenging. As in dataset 1, we simulated a $10{,}000 \times 784$ matrix $\boldsymbol{X}$ whose element $x_{ij} \sim$ i.i.d. $U(0,1)$ and divided the samples into two equal-size classes $C_1$ and $C_2$. But then, to make $p' = 64$ randomly chosen variables significant, we let $x_{ij} \leftarrow x_{ij} + (2\alpha_{ij} - 1) \times \delta_{ij}$ for $i \in C_1$, $j \in \Omega_{p'}$, where $\alpha_{ij} \sim$ Bernoulli($\frac{1}{2}$), and $\delta_{ij} \sim U(0.8, 1)$. Note that different from the first two simulation schemes, here $\alpha$ and $\delta$ depend on both $i$ and $j$.

For dataset 4, the data matrix was $\boldsymbol{X} = (x_{ij})_{10{,}000 \times 784}$, and each $x_{ij} \sim U(-1,1)$. Of the 784 variables, 64 were randomly chosen as significant variables (denoted by $x_{k_j}, j = 1, \ldots, 64$), and $y$ depended on the main effects and interactions of $x_{k_j}$ as well as their nonlinear transformations:

$$y_i = \sum_{j=1}^{16} \beta_j x_{ik_j} + \sum_{j=17}^{32} \beta_j \sin x_{ik_j} + \sum_{j=33}^{48} \beta_j e^{x_{ik_j}} + \sum_{j=49}^{64} \beta_j \max(0, x_{ik_j}) + \beta'_1 x_{ik_{15}} x_{ik_{16}} + \beta'_2 x_{ik_{31}} x_{ik_{32}} + \beta'_3 x_{ik_{47}} x_{ik_{48}} + \beta'_4 x_{ik_{63}} x_{ik_{64}} + \varepsilon_i,$$

where $\beta_j = (2\alpha_j - 1) \times b_j$, $\alpha_j \sim$ Bernoulli($\frac{1}{2}$), $b_j \sim U(1,3)$, $\varepsilon_i \sim N(0,1)$ for $i = 1, \ldots, 10{,}000$, $j = 1, \ldots, 64$, and $\beta'_1$, $\beta'_2$, $\beta'_3$, $\beta'_4$ had the same distribution as $\beta_j$.

*Implementation details.* Except for the MNIST data, which contain 60,000 training images (including 5,000 validation images) and 10,000 testing images, each dataset was divided into a training set and a test set, with 80% of the samples in the training set and 20% in the test set, and 30% of training samples were further separated for validation (used to decide when to stop training, see Supplementary Information).

SurvNet was implemented on TensorFlow 1.8[44]. We used a common network structure for all datasets, which had two hidden layers consisting of 40 and 20 nodes respectively. The ReLU activation function was used, together with a batch size of 50 and a learning rate of 0.05 (0.01 for the regression problem).

## Data availability

The simulated data (datasets 1–4) were generated using the code at https://github.com/zixuans/SurvNet/tree/master/Data. The MNIST data (dataset 5) is available at http://yann.lecun.com/exdb/mnist/. The single-cell RNA-Seq data (dataset 6) is available at the GEO repository https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE87544. The synthetic data used in the NeurIPS paper[16] were simulated using the code on https://github.com/zixuans/SurvNet/tree/master/Comparisons%20with%20knockoffs/Scenario%203, and the real datasets were provided by request from its author, Y. Lu. The synthetic data used in the AISTATS paper[17] were simulated using the code at https://github.com/zixuans/SurvNet/tree/master/Comparisons%20with%20knockoffs/Scenario%204, and the two real datasets are available at https://archive.ics.uci.edu/ml/datasets/Bank+Marketing and https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data.

## Code availability

The code developed for the study of SurvNet is publicly available at the Github repository https://github.com/zixuans/SurvNet. The code for GL and SGL[13] is publicly available at https://bitbucket.org/ispamm/group-lasso-deep-networks/src/master/. The code used to construct second-order knockoffs[19] and deep knockoffs[36] is available at https://github.com/msesia/knockoff-filter and https://github.com/msesia/deepknockoffs, respectively. The code of the algorithm proposed in the AISTATS paper[17] is publicly available at https://github.com/jroquerogimenez/ConditionallySalientFeatures.

Received: 25 July 2019; Accepted: 26 January 2021;
Published online: 29 March 2021

## References

1. May, R., Dandy, G. & Maier, H. Review of input variable selection methods for artificial neural networks. *Artif. Neural Networks* **10**, 16004 (2011).
2. Guyon, I. & Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003).
3. Chandrashekar, G. & Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **40**, 16–28 (2014).
4. Battiti, R. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Networks* **5**, 537–550 (1994).
5. May, R. J., Maier, H. R., Dandy, G. C. & Fernando, T. G. Non-linear variable selection for artificial neural networks using partial mutual information. *Environ. Model. Software* **23**, 1312–1326 (2008).
6. Maier, H. R., Dandy, G. C. & Burch, M. D. Use of artificial neural networks for modelling cyanobacteria *Anabaena* spp. in the River Murray, South Australia. *Ecol. Model.* **105**, 257–272 (1998).
7. Brill, F. Z., Brown, D. E. & Martin, W. N. Fast generic selection of features for neural network classifiers. *IEEE Trans. Neural Networks* **3**, 324–328 (1992).

8. Tong, D. L. & Mintram, R. Genetic Algorithm-Neural Network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection. *Int. J. Mach. Learn. Cybern.* **1**, 75–87 (2010).
9. Sivagaminathan, R. K. & Ramakrishnan, S. A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert Syst. Appl.* **33**, 49–60 (2007).
10. Grandvalet, Y. & Canu, S. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In *Advances in Neural Information Processing Systems* 445–451 (1999).
11. Chapados, N. & Bengio, Y. Input decay: simple and effective soft variable selection. In *IJCNN'01. International Joint Conference on Neural Networks* Vol. 2, 1233–1237 (IEEE, 2001).
12. Similä, T. & Tikka, J. Combined input variable selection and model complexity control for nonlinear regression. *Pattern Recognit. Lett.* **30**, 231–236 (2009).
13. Scardapane, S., Comminiello, D., Hussain, A. & Uncini, A. Group sparse regularization for deep neural networks. *Neurocomputing* **241**, 81–89 (2017).
14. Zhang, G. P. Neural networks for classification: a survey. *IEEE Trans. Syst. Man Cybernet. C* **30**, 451–462 (2000).
15. Benjamini, Y. & Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Stat. Soc. B* **57**, 289–300 (1995).
16. Lu, Y., Fan, Y., Lv, J. & Noble, W. S. Deeppink: reproducible feature selection in deep neural networks. In *Advances in Neural Information Processing Systems* 8676–8686 (2018).
17. Gimenez, J. R., Ghorbani, A. & Zou, J. Knockoffs for the mass: new feature importance statistics with false discovery guarantees. In *22nd International Conference on Artificial Intelligence and Statistics* 2125–2133 (2019).
18. Barber, R. F. & Candès, E. J. Controlling the false discovery rate via knockoffs. *Ann. Stat.* **43**, 2055–2085 (2015).
19. Candès, E., Fan, Y., Janson, L. & Lv, J. Panning for gold: 'model-X' knockoffs for high dimensional controlled variable selection. *J. R. Stat. Soc. B* **80**, 551–577 (2018).
20. Storey, J. D. & Tibshirani, R. Statistical significance for genomewide studies. *Proc. Natl Acad. Sci. USA* **100**, 9440–9445 (2003).
21. Benjamini, Y. & Yekutieli, D. The control of the false discovery rate in multiple testing under dependency. *Ann. Stat.* **29**, 1165–1188 (2001).
22. Heesen, P. et al. Inequalities for the false discovery rate (FDR) under dependence. *Electron. J. Stat.* **9**, 679–716 (2015).
23. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
24. Kolodziejczyk, A. A., Kim, J. K., Svensson, V., Marioni, J. C. & Teichmann, S. A. The technology and biology of single-cell RNA sequencing. *Mol. Cell* **58**, 610–620 (2015).
25. Chen, R., Wu, X., Jiang, L. & Zhang, Y. Single-cell RNA-Seq reveals hypothalamic cell diversity. *Cell Rep.* **18**, 3227–3241 (2017).
26. Romano, Y., Sesia, M. & Candès, E. Deep knockoffs. *J. Am. Stat. Assoc.* **115**, 1861–1872 (2019).
27. Tetko, I. V., Villa, A. E. & Livingstone, D. J. Neural network studies. 2. Variable selection. *J. Chem. Inf. Comput. Sci.* **36**, 794–803 (1996).
28. Steppe, J. & Bauer, K. Jr Feature saliency measures. *Comput. Math. Appl.* **33**, 109–126 (1997).
29. Sen, T. K., Oliver, R. & Sen, N. in *Neural networks in the Capital Markets* 325–340 (Wiley, 1995).
30. Yacoub, M. & Bennani, Y. HVS: A heuristic for variable selection in multilayer artificial neural network classifier. In *Intelligent Engineering Systems Through Artificial Neural Networks, St. Louis, Missouri* Vol. 7, 527–532 (1997).
31. Garson, D. G. Interpreting neural network connection weights. *AI Expert* **6**, 47–51 (1991).
32. Nath, R., Rajagopalan, B. & Ryker, R. Determining the saliency of input variables in neural network classifiers. *Comput. Oper. Res.* **24**, 767–773 (1997).
33. Gevrey, M., Dimopoulos, I. & Lek, S. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecol. Model.* **160**, 249–264 (2003).
34. Mozer, M. C. & Smolensky, P. Skeletonization: a technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems* 107–115 (1989).
35. Karnin, E. D. A simple procedure for pruning back-propagation trained neural networks. *IEEE Trans. Neural Networks* **1**, 239–242 (1990).
36. LeCun, Y., Denker, J. S. & Solla, S. A. Optimal brain damage. In *Advances in Neural Information Processing Systems* 598–605 (1990).
37. Cibas, T., Soulié, F. F., Gallinari, P. & Raudys, S. Variable selection with optimal cell damage. In *International Conference on Artificial Neural Networks* 727–730 (Springer, 1994).
38. Hassibi, B. & Stork, D. G. Second order derivatives for network pruning: optimal brain surgeon. In *Advances in Neural Information Processing Systems* 164–171 (1993).
39. Dimopoulos, Y., Bourret, P. & Lek, S. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Process. Lett.* **2**, 1–4 (1995).
40. Dimopoulos, I., Chronopoulos, J., Chronopoulou-Sereli, A. & Lek, S. Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in Athens city (Greece). *Ecol. Model.* **120**, 157–165 (1999).
41. Ruck, D. W., Rogers, S. K. & Kabrisky, M. Feature selection using a multilayer perceptron. *J. Neural Network Comput.* **2**, 40–48 (1990).
42. Bishop, C. M. et al. *Neural Networks for Pattern Recognition* (Oxford Univ. Press, 1995).
43. LeCun, Y. A., Bottou, L., Orr, G. B. & Müller, K.-R. in *Neural Networks: Tricks of the Trade* 9–48 (Springer, 2012).
44. Abadi, M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems (2015); https://www.tensorflow.org/


## Acknowledgements

This work was supported by the National Institutes of Health (R01GM120733 to J.L.), the American Cancer Society (RSG-17-206-01-TBG to J.L.) and the National Science Foundation (1925645 to J.L.).

## Author contributions

J.L. conceived and supervised the study. J.L. and Z.S. proposed the methods. Z.S. implemented the methods and constructed the data analysis. Z.S. drafted the manuscript and J.L. substantively revised it.

## 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/s42256-021-00308-z.

**Correspondence and requests for materials** should be addressed to J.L.

**Peer review information** *Nature Machine Intelligence* thanks the anonymous reviewers for their contribution to the peer review of this work.

**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.