nature portfolio

https://doi.org/10.1038/s42256-022-00525-0

Supplementary information

Deep neural networks with controlled variable selection for the identification of putative causal genetic variants

In the format provided by the authors and unedited

Supplementary Information

1. Notes on the real data preparation

We focus on both common (MAF $\geq$ 1%) variants and rare (MAF < 0.01, MAC $\geq$ 10) genetic variants imputed at high resolution based on the reference panels from TOPMed using the Michigan Imputation Server[39]. Five knockoff variants were generated for each variant. We additionally adjusted the learner for sex and 10 leading principal components as covariates. For task A, the validation AUC for Lasso-MK is 0.734, for stabilized HiDe-MK is 0.73, and for DeepPINK-MK with ReLU activation function is 0.709; For task B, the validation AUC for Lasso-MK is 0.731, for stabilized HiDe-MK is 0.7337. The reported AUC values reflect prediction accuracy of genetic variants, sex, and PCs, based on a 5-fold cross-validation. We report results based on target FDRs 0.05, 0.10 and 0.20. Each identified genetic variant is then assigned to either a gene or an intergenic region. If the variant is within a gene, we report the gene's name and if it is in an intergenic region, we report the upstream and downstream genes.

Identity-by-descent was run to determine the relatedness between all individuals using PLINKv1.9 [53]. We filtered for duplicates and kept the copy from the SNP array with the highest genome coverage and removed first degree relatives keeping AD relatives over controls; and when both had a concordant diagnosis, we kept the younger case or older control. On the subset of remaining individuals, we computed genetic principal components to account for population stratification[54].

Prior to ancestry determination, SNPs were filtered based on genotyping rate (< 0.95), MAF < 0.01 and Hardy-Weinberg equilibrium (HWE) in controls ($p < 10^{-6}$). By applying an ancestry percentage cut-off > 0.75, the samples were stratified into five super populations such as South-Asians, East-Asians, Americans, Africans and Europeans, and an Admixed group composed of individuals not passing cut-off in any single ancestry. To avoid spurious associations as the most individuals were Europeans, we restricted our analysis to European ancestry individuals. Then, we did an individual-level ancestry determination for each cohort included in our analysis with SNPWeights v2.1[55] using reference populations from the 1000 Genomes Consortium[56]. For further genetic quality control and imputation, we used the gnomAD database[57] to filter out SNPs that met one of the following criteria: (i) located in low complexity region, (ii) located within common structural variants (MAF > 0.01), (iii) multiallelic SNPs with MAF > 0.01 for at least two alternate alleles, (iv) located within a common Ins/Del (insertion/deletion), (v) having any flag different than PASS in gnomAD. Individuals with more than 0.05 genotype missingness were excluded. Imputation was performed on the Michigan imputation server using the TOPMed reference panel[39,58]. Per cohort, only variants with sufficient imputation quality ($r^2 > 0.3$) were included in the analysis.

2. Model configurations

We search for optimal hyper-parameters of HiDe-MK through a randomized search with a 5-fold cross validation. In our search space for *simulation data*, we have 200 sets of hyper-parameters with epoch number set to 200. We do this for every replicate of 500 replicates and for both dichotomous and quantitative traits. For L-1 norm values of DeepPINK and stabilized HiDe-MK, we used the default lambda path same as the way it is generated in GLMNET library of R. We used batch size 1024, a learning rate 0.001, the number of filters 8, and the kernel size for the 1st layer of locally connected layer is set to 6 and for the 2nd layer of locally connected layer is set to 25. FIs are extracted with running on the whole data and for every epoch and every set of hyper-parameters. Then, we apply knockoff statistics and measure FDR and Power. For SVM and SVR learners, we used the LiblineaR, an R package for the predicative linear models (https://cran.r-project.org/web/packages/LiblineaR/index.html) for support vector classification

and regression with linear kernels. We used the default parameter to set the tolerance criterion for SVM and SVR optimization which is 0.01. We used the primal objective function with L2-loss support vector regression. When a primal solution is found, we extract feature importance scores. For SVM, this package uses a linear kernel same as the one in SVMlight which is a C++ implementation of SVM, embedded into this R library. This linear SVM has a parameter cost Q which is the cost of constraints violation in SVM objective function. We adjust this parameter to be in the range Q = {1000, 500, 100, 10, 1, 0.1, 0.01, 0.05, 0.001}. This parameter rules the trade-off between the regularization and correct classification. For Ridge regression and Lasso regression, we used the R package GLMNET. The tuning parameter λ was tuned by the default lambda path and a five-fold cross-validation.

For the *real data analysis*, we trained stabilized HiDe-MK up to 300 epochs which lasts about 15h with GPU applied to the imputed data. The batch size is set to 512 which we found is sufficient for fast training without getting out of memory error. The number of filters was 8. The kernel size for the 1st layer of locally connected layer is set to 6 and for the 2nd layer of locally connected layer, for our simulation studies, it is set to 5, and for real data analysis, we adjust this parameter in the range {200, 250, ..., 1000}. The number of neurons in the dense layer was set to 50. With this setup, the final size of stabilized HiDe-MK or the total number of weight parameters with kernel size 500 in the second locally connected layer was 172,930. We only used 200 values of a L-1 norm regularizer for the first locally connected layer. The proposed method is independent of the choice of optimal epoch number. The activation function was 'ELU' for every layer except the first and the last layer. For the decision layer, we used Sigmoid activation function for dichotomous trait and linear activation function for the continuous trait in simulation studies and real data analysis. To do experiments, we used Sherlock high-performance computing (HPC) device at Stanford University. The configurations for cluster of remote machines were GPU: Nvidia GeForce RTX 2080 Ti 11GB; and CPU: Intel Xeon(R) Silver 4116 2.10GHz; 128GB of RAM (8 cores), with OS: Ubuntu16.04.3 LTS.

3. Additional figures and experiments

Figure S1. Architecture of HiDe-MK for real data analysis. The framework is applied to a data with 11,662 genetic variants with five sets of knockoffs; every knockoff cohort has the same size as the original cohort. The covariates with 11 features are fed to the neural network right before the last layer. The genotype data is presented to the network as the first input data with size 11662×6 where 6 is number of original and knockoff features. The term "?" is referred to the batch size, and in our experiments, we set it to 512. In the next layer, each 6 neurons, i.e., one original feature and its 5 knockoffs are grouped together. The shape of matrix changes from 11662×6 to 11662×1. Next, every 500 neurons are grouped and are further mapped to 8 channels. Doing so, the shape of data changes from 11662×1 to 24×8. Then a dropout layer with 0.10 dropout rate is applied and the two-dimensional data is then flattened from the size 24×8 to 192×1. These 192 neurons are passed to a dense layer with 50 neurons. The 11 covariates are then concatenated to these 50 neurons. Note that the activation function for covariate layer is linear. The last layer with one neuron is the decision layer. This whole process is repeated for the whole samples of data till the total number of epochs are completed.

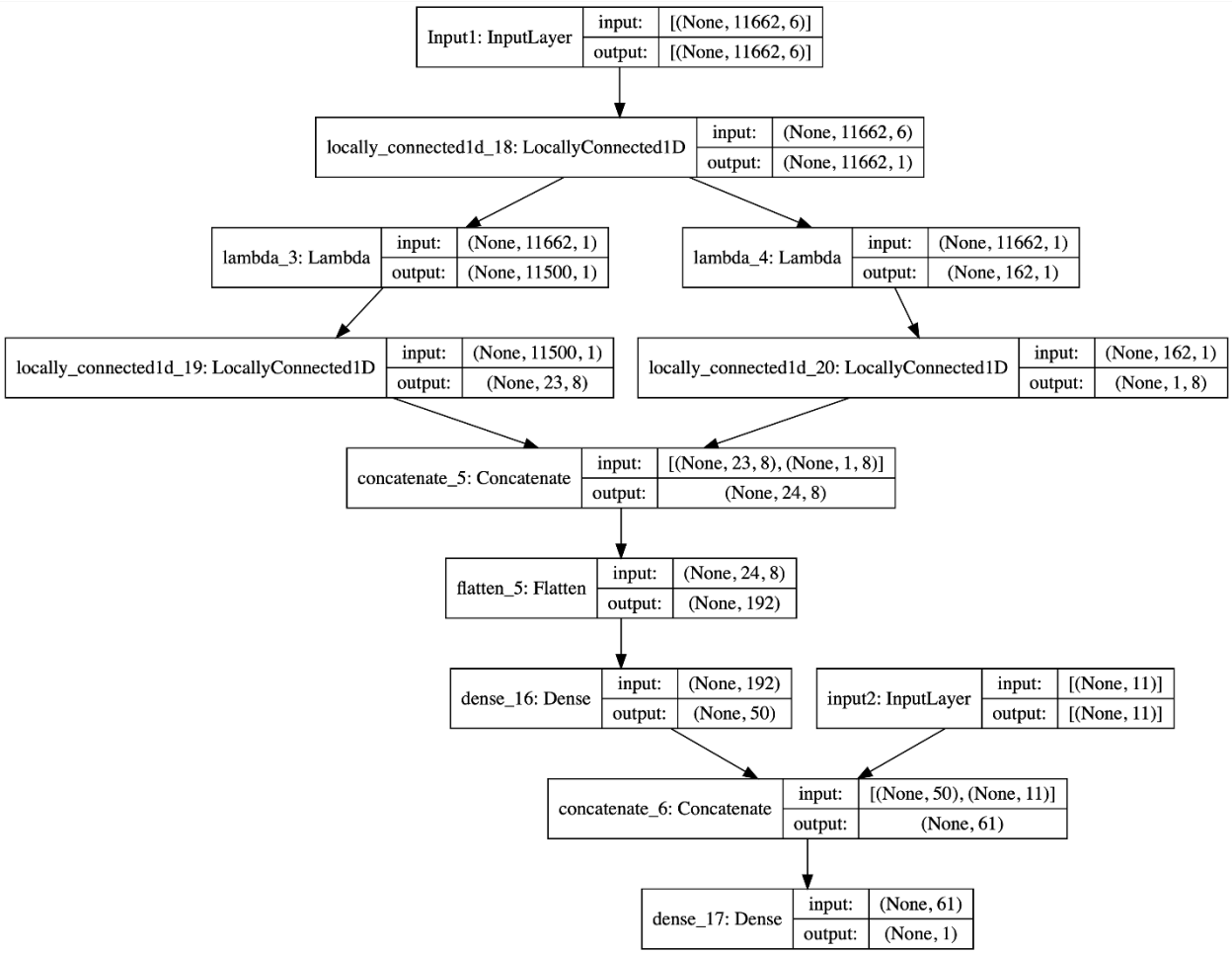

Figure S2. The behavior of different activation functions applied to both dichotomous and continuous traits. Activation functions ELU, ReLU, Swish, and GeLU equipped with stabilized HiDe-MK are applied to SKAT data with dichotomous and continuous traits, validated over 200 replicates. The search space for hyper-parameter tuning is the same as what we have done earlier with ELU activation function. We used 100 different combination of hyper-parameters and 200 epoch numbers.

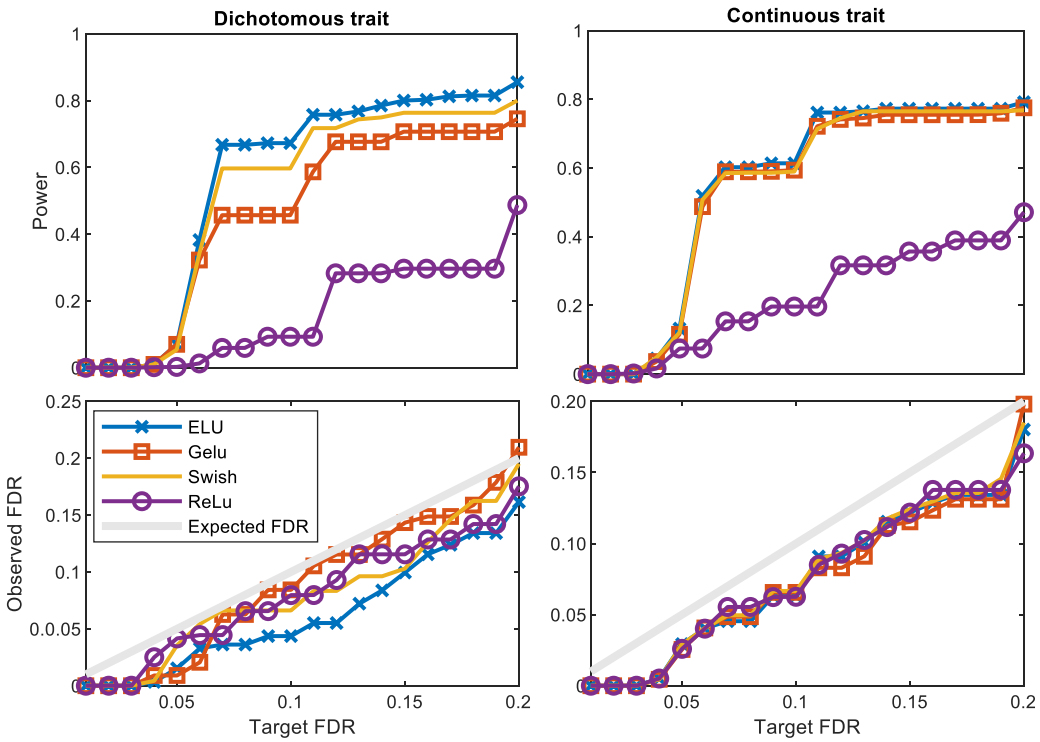

Figure S3. Empirical FDR and power reported for different learning methods at different target FDR values applied to simulation studies, both dichotomous and continuous traits. All methods are equipped with single knockoff.

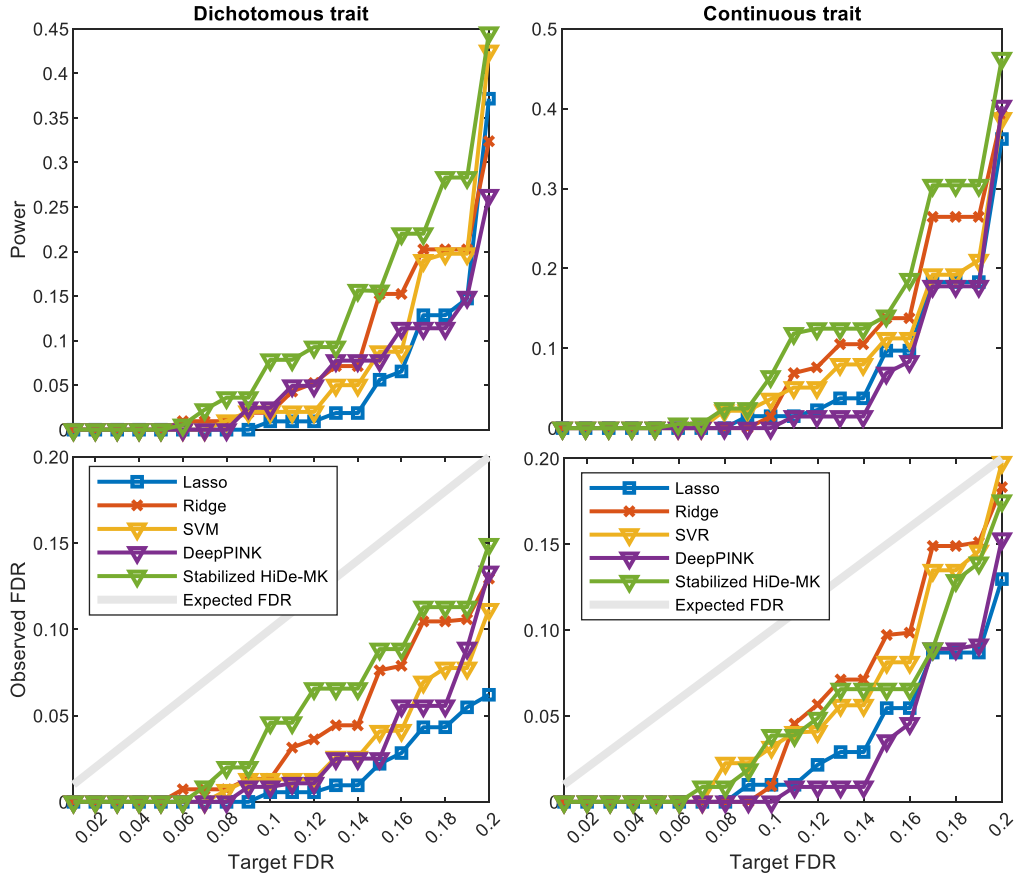

Figure S4. Results of FDR control for dichotomous and quantitative traits without minor allele count filtering.

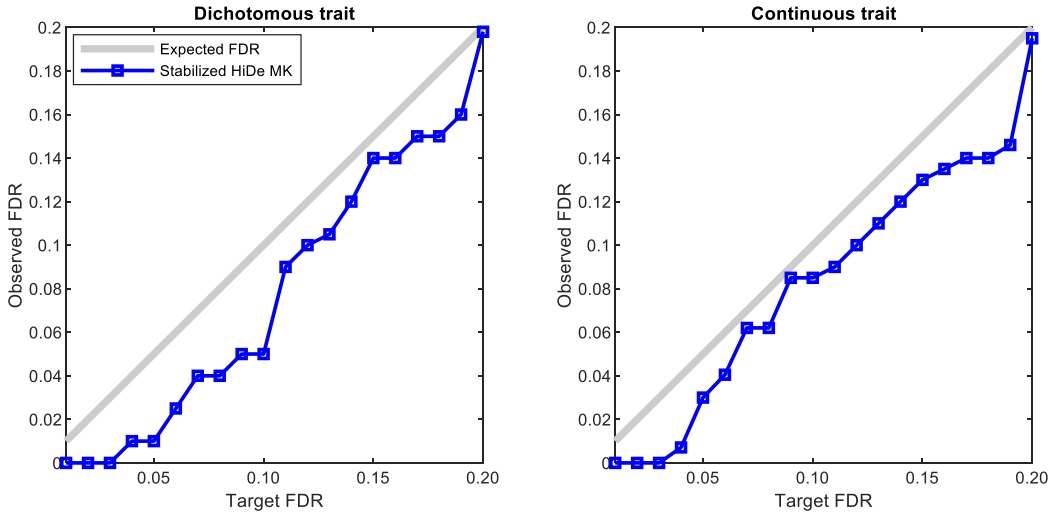

Figure S5. Manhattan plot for a deepPINK with ReLU activation function and with multiple knockoffs. Each dot point represents a genetic variant. The dashed horizontal lines indicate target FDRs 0.05, 0.10 and 0.20. DeepPINK-MK suffers from dying ReLU problem and hence the learning convergence is very slow, and the validation AUC is poor (0.73 for the stabilized HiDe-MK with ELU versus 0.709 for deepPINK-MK). Because of this, only 19 genetic variants, associated with 7 genes could be identified by this method, only in chromosome numbers 2 and 19.

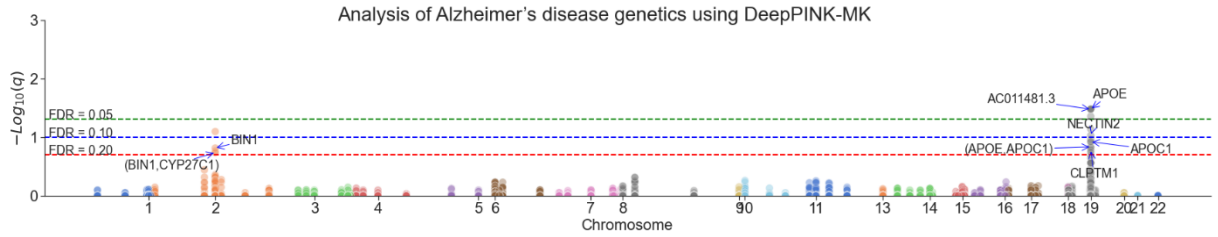

Figure S6. **Manhattan plots for a marginal test**. Each dot point represents a genetic variant. The dashed horizontal line indicates the conventional threshold value 5e-8.

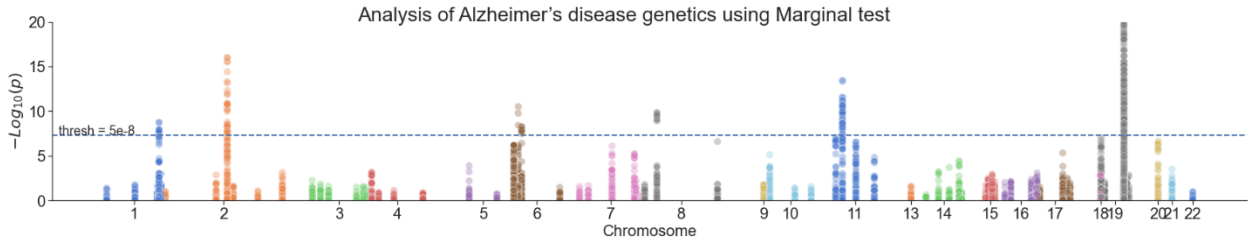

Figure S7. Impact of different kernel size values on measured FDR, power, and the total number of weight parameters. The effectiveness of the locally connected layers in the architecture of HiDe-MK depends on the proper choice of the kernel size, i.e., the number of neurons that should be grouped. To empirically validate its importance on HiDe-MK performance, we ran it on a dichotomous trait with different kernel size values and measured the average FDR, power, and the size of network over 50 runs. The search space for hyperparameters of HiDe-MK is same as we discussed in Results section and DNN configurations in Methods section. The learner with kernel size 25 has superior power while FDR is under control and the number of weight parameters are much less than when it is compared to the small kernel size 5.

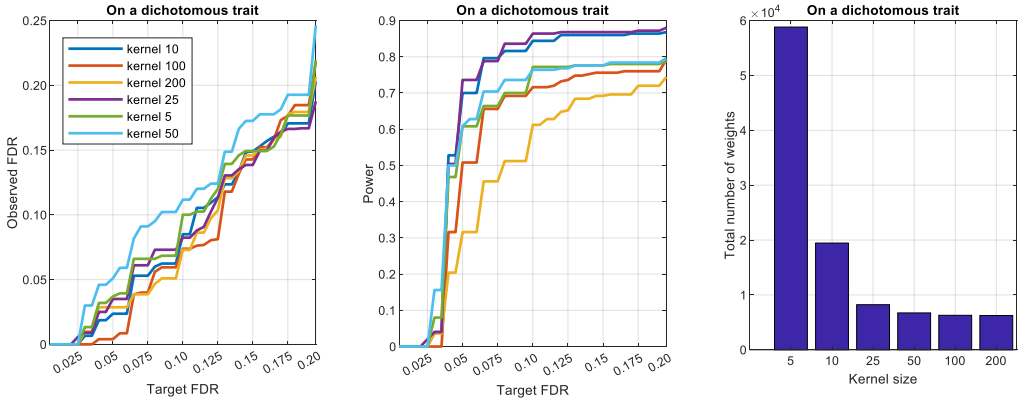

Table S1. Empirical power at target FDR 0.2 based on 500 replicates and standard deviation of the power estimation for different learners.

	Dichotomous trait		Continuous trait	
	Power	Standard deviation	Power	Standard deviation
Lasso	0.6795	0.0127	0.6725	0.0149
Ridge	0.4999	0.0152	0.5145	0.0183
DeepPINK	0.3994	0.0184	0.4340	0.0199
SVM	0.6819	0.0152	0.6708	0.0169
Stabilized HiDe-MK	0.8392	0.0089	0.7839	0.0072

Table S2. List of identified genes for both Lasso-MK and stabilized HiDe-MK at target FDR 0.10 for the confirmatory stage analysis of existing AD risk variants.

	Identified Genes
Lasso-MK	*BIN1, CYP27C1, CD2AP, NYAP1, CLU, AL512631.1, APOE, AC011481.3, APOC1, SHARPIN, MS4A4A, CR1, MEF2C-AS1, TREM2, TREML2, ABCA7, CLNK, HLA-DRB1, HLA-DQA1, CASS4, USP6NL, ECHDC3, RNU6-560P, LINC02695, TSPOAP1, ABCA7*
Stabilized HiDe-MK	*BIN1, CYP27C1, CD2AP, TOMM40, CLU, APOE, AC011481.3, APOC1, SHARPIN, MS4A4A, CR1, ABCA7, KAT8, CLPTM1, ADGRF, TREM2, TREML2, MS4A4E, NECTIN2, AC011481.2, CASTOR3, RNF111, SLTM, EPHA1, CASS4, USP6NL, ECHDC3*

Table S3. List of identified genes for both Lasso-MK and stabilized HiDe-MK at target FDR 0.10 for the functionally informed analysis of pQTLs.

	Identified Genes
Lasso-MK	*BIN1, AC110926.1, AC245884.2, APOE, CR1, MS4A6A, PILRA, PLCG2, SPI1*
Stabilized HiDe-MK	*AC110926.1, AC011481.3, APOC1, APOE, BIN1, CUX1, KCNN4, MCM6, MUC12, NCR2, PILRA, PLAUR, PRH1, SIGLEC18P, TRIM56, TREM1, TREML2*

Table S4. **The commonly used activation functions in the layers of DNN**. the underlying structure of data can be learned through linear or nonlinear activation functions.

Activation function	Mathematical equation
ReLU	Max[z,0]
ELU	$f(z) = \begin{cases} z & if\ z > 0 \\ a(e^z - 1) & otherwise \end{cases}$
Sigmoid	$(1 + e^{-z})^{-1}$
Tanh	$(e^z - e^{-z})(e^z + e^{-z})^{-1}$
Linear	Z

4. Pseudo code for stabilized HiDe-MK

In the following, we present the pseudo code for running the proposed stabilized HiDe-MK method in the simulation studies. Codes have been written in Python with Keras and TensorFlow using the libraries "Numpy", "Pandas", "SKlearn", "csv", "time", "os" and "sys". We used R program for the simulation data generation and knockoff inference to obtain q-values, FDR, and Power. We used R libraries "SKAT", "knockoffs", "KnockoffScreen" and "data.table". We directly imported the functions "MK.threshold.byStat" "MK.statistics", "MK.threshold" and "MK.q.byStat" from the KnockoffScreen software package to our R script.

This whole package has been compiled to work with Python 3.6, Tensorflow 2.6, Keras 2.6, and R version 3.6 on Windows 10 and Mac OS BigSur. We also successfully ran the package on Stanford supercomputer Sherlock remotely. We made the package publicly available at Github with the link: {https://github.com/Peyman-HK/De-randomized-HiDe-MK} which is licensed under General Public license 3.0 (GPL-3.0) with a descriptive manual for demo instructions, for running the code, explained in the Readme.MD file in Github. The user only needs to install necessary libraries and download the whole package and change the necessary directories to read the data.

In the following, we show four pseudo-codes for applying the stabilized HiDe-MK on simulation datasets from the data generation to the obtained FDR and Power.

Pseudo-code 1

```
  # -- Generate original and knockoff features for simulation data --
  # -- Original data generation --
  Input: Given parameters for data generation
  Output: The original matrix G and knockoff matrices GKs
  (Generate data (through SKAT R library) by the calibration coalescent model
(COSI) with mimicking LD structure of European ancestry)
  # -- Initialize parameters --
  let Num_samples = 10000, Num_features = 2000
```

 # Generate SNP sets with initialized parameters

```
  1- Through COSI model, and given sample and feature size, generate genotype
data with their positions, which has the size 10000 × 2000.
  2- Obtain MAC - summation for each column of SNP.set.
```

 # Quality control

3 Filter out ultra-rare variants with MAC > 10
4- Remove columns with identical values to obtain the original matrix G.

-- **Knockoff data generation** through KnockoffScreen algorithm --
Input: Position for genetic variants, number of knockoffs, original data G
Output: M knockoff data matrices GKs

1- Extract the positions for variants (assuming that variants are on the same chromosome; if not, apply knockoff generation to each chromosome separately)
2- Set number of knockoffs to M = 5
3- Generate knockoff through the function create.MK(G, pos, M=5)

Pseudo-code 2

```
# Stabilized HiDe-MK algorithm to learn from data and obtain ensemble of gradients or feature importance scores (FIs)
Input: Both original and knockoff datasets and the covariates data: X_G = [G, G_Ks] and covariates X_P, target vector Y, different combinations of hyper-parameters (Num_param)
Output: a single FIs which is aggregation of all FIs

For Param in Num_param
     For Epoch in Num_epoch

     1- Train HiDe-MK with current hyper-parameters through 5-fold CV
     Trained_model = Train_HiDe([X_G_train, X_P_train], Y_train)
     2- Validate HiDe-MK with current hyper-parameters through 5-fold CV
     Y_predict_test = Trained_model([X_G_test, X_P_test], Y_test)
     3- Calculate validation AUCs
     Validation_Loss{Param,Epoch} = Loss_measure(Y_test, Y_predict_test)

     END
END

4- Obtain the average of Loss values for every epoch and hyper-parameter
Average_Loss = mean(Validation_Loss)
5- Based on averaged losses, assign weights to every epoch and parameter
W_all = Average_Loss / min(Average_Loss)

# Obtain FIs on the whole data (G_All) for every epoch and hyper-parameter
For   Batch in {G_All}

      6- Calculate predicted values for the current batch
      predicted_batch = NN_predict(Current_batch, Y_Batch)
      7- Find FIs for each batch through the GradientTape function
      FIs{Batch} = GradientTape(G_All(Batch), predicted_batch)
```

END

8- Stack FIs obtained for each batch to get the FIs for that epoch number

9- Aggregate all FIs following and obtain a single weighted FIs:
Aggregated_FI = {FI(epoch, param) × W(epoch,param)}/ sum(W_all)

Pseudo-code 3

 # Knockoff Inference to obtain FDR and Power through FIs
Input: Obtained FIs through learned HiDe-MK
Output: Empirical FDR and Power at different levels of target FDR
(Through FIs, Kappa, Tau and then q-values are obtained, and then empirical FDR and Power are measured at different target FDR levels)

let FDR_levels = {0.01, 0.02,…,0.20}

1- Obtain Kappa and Tau through function MK.statistic with original FIs and knockoffs FIs. T_0 is the FIs for original features and T_K is the FIs for knockoffs.
Kappa_Tau = MK.statistic(T_0,T_k,method='median')
2- Obtain q-values through the function MK.q.byStat
q_vals =MK.q.byStat(Kappa,Tau,M)

Apply knockoff filter for different FDR levels
For values in FDR_levels:

3- Select Features which have a q-value <= a target FDR
4- Find empirical FDR and Power comparing candidates and true signals

End