SUPPORTING INFORMATION

Clinically Applicable Deep Learning Algorithm Using Quantitative Proteomic Data

Hyunsoo Kim[1,2,3#], Yoseop Kim[3#], Buhm Han[2], Jin-Young Jang[4*], and Youngsoo Kim[1,2,3*]

[1]Institute of Medical and Biological Engineering, Medical Research Center, [2]Departmet of Biomedical Sciences, [3]Departmet of Biomedical Engineering and [4]Department of Surgery, Seoul National University College of Medicine, Yongon-Dong, Seoul 110-799, Republic of Korea

[#] The first 2 authors contributed equally to this work.

* Corresponding authors:

Jin-Young Jang, M.D., Ph.D.

Department of Surgery

Seoul National University College of Medicine

103 Daehak-ro, Seoul 03080, Republic of Korea

Tel: 82-10-8338-6719, E-mail address: jangjy4@snu.ac.kr

and

Youngsoo Kim, Ph.D.

Department of Biomedical Engineering

Seoul National University College of Medicine

103 Daehak-ro, Seoul 03080, Republic of Korea

Tel: 82-10-5351-1959, E-mail address: biolab@snu.ac.kr

SUPPLEMENTARY METHODS:

Hyperparameter optimization of deep learning method.

SUPPLEMENTARY TABLES:

SUPPLEMENTARY FIGURES:

SUPPLEMENTARY METHODS

Hyperparameter optimization of deep learning method

The R script for all processing steps of the deep learning (DL) method followed the single document (1). We have included all preprocessing and learning steps of the DL method as an R script in Supporting Information.

Epochs

Epoch is a parameter that comprises the number of repetitions in the entire DL network during learning. Generally, it is important to find the appropriate number of epochs, because execution time and overfitting problems are associated with epochs. We tested this parameter in 10 10-epoch intervals from 1 to 100 and 100-epoch intervals from 100 to 1000. The model performed best at an epoch of 400, and the performance started to deteriorate at over 400 times (Figure 2A), likely due to overfitting. Because the number of epochs increased in DL, the model was fitted for the training dataset rather than the independent test dataset—ie, overfitting occurred when the generalization capability of the model was low. We confirmed that many epochs do not always produce good-performance models.

Nodes (neurons) and hidden layers

A neuron has a function that takes multiple numerical inputs and generates 1 numerical output. These neurons are organized into layers, and the outputs from all neurons in one layer

become the inputs for each neuron in the next layer. The number of neurons is generally calculated by multiplying the input size by two-thirds plus the output size—not exceeding twice the input size. By applying these rules to 34 inputs (features), we evaluated models in 20, 40, and 60 neurons. For hidden layers, we only applied 1 hidden layer, but it performed better than traditional machine learning methods (Table S1, No. 22-24). We found that stacking the hidden layer up to 3 did not make a significant difference with regard to performance. The best performance was observed at 20 neurons and 2 hidden layers (Table S2, No. 22–24 & Figure S1A).

Activation function

Activation functions are the most important feature of DL, giving the network nonlinearity. Due to the characteristics of these nonlinear functions, DL identifies relationships between nonlinear data, as well as linear relationships (2). H2O supports 3 activation functions: Rectified linear unit (ReLU, rectifier), Tanh, and Maxout. Rectifier is generally known to perform well in most cases, is the default and most common activation function, and has a quicker execution time than other activation functions. It outputs the sum of its weighted inputs but clips all negative values to 0. Tanh is short for 'hyperbolic tangent.' This function takes an input range of negative infinity to positive infinity and converts it to an output range of -1 to +1. But, it varies most rapidly when the sum of the inputs is close to 0. Maxout simply outputs the highest value (the max) of the inputs meaning that the weighted inputs are used directly and not summed. Our results show that rectifier had the best performance and the shortest execution time (Table S2, No. 15, 30, and 31 & Figure 2B).

Rho and epsilon

Rho adjusts the learning rate, and epsilon adjusts the learning decay. Rho relates the memory to the prior weight update. In general, rho is between 0.900 and 0.999, and the default value is 0.990. Typically, epsilon is between 1E-4 and 1E-10, and the default value is 1E-8. Epsilon is recommended for testing from a large value to a smaller value. We tested each condition and noted the best classification at a rho of 0.990 and an epsilon at 1E-8 (Figures S1B & S2C).

L1 and L2 regularization

To prevent model overfitting, H2O supports regularization and dropout parameters. L1 (also called lasso regularization or lasso regression) and L2 regularization (also called ridge regularization or ridge regression) increase the stability of the model through penalty to cover many weights and penalty to prevent weight enlargement, respectively. L1 generally works well when there are a lot of noisy data. L2 prevents 1 weight value from predominating and keeps all weight values small. Both have a default value of 0, and typical values are 1E-4 or smaller. We tested L1 and L2 at 0, 1E-4, and 1E-5. The classification performance was best when L1 was 0 and L2 was 1E-5 and when only L2 was applied (Figure S1C).

Dropout (input dropout ratio & hidden dropout ratio)

Dropout parameters are also used to prevent overfitting of the model in DL. Dropout randomly set some nodes to 0 in the forward pass. H2O supports 2 types of dropout parameter settings: input dropout ratio and hidden dropout ratio. Input dropout ratio is the percentage of input

nodes that are fed into the first hidden layer. At 0.5, there is a 50% chance of each feature being used for each row in the training dataset. The default value is 0.00, and we tested the range from 0.00 to 0.50. We specified 1 dropout ratio per hidden layer. The default value of 0.5 indicates that for each training row that is processed through the network, there is a 50% chance that a neuron will pass its value on to the next hidden layer and a 50% chance that it will pass on 0. We tested at hidden dropout ratios of 0.5 and 0.6. The dropout of the input layer was not effective, and the performance improved when a ratio of 0.5 was assigned to the hidden layer (Figure 2D & Figure S1D).

Training samples per iteration

Training sample per iteration is the number of training rows (samples) that are used per iteration. The value of -2 is the default that activates the automatic mode (auto-tuning), lets *H2O* decide the training samples per iteration. The best performance was seen at -2. The special values are 0 for 1 epoch per iteration and -1 for processing the maximum amount of data per iteration. (Figure S1E).

Max w2

Max w2 is the upper limit for the (squared) sum of the incoming weights to a neuron. This is a direct way to stop weights from growing too large to model. We tested at max w2 values of 0 and 10. The model performance was higher when max w2 was 10 versus 0 (Figure S1F).

The DL method as an R script

```
#Load libraries

library(h2o)

library(pROC)

library(preprocessCore)

library(ggplot2)

localH2O = h2o.init()


#Load the dataset

dataset_df <- read.csv(file="file_name.csv",check.names = T)

dataset_df <- dataset_df[,-1]

dataset_df$group <- as.factor(ifelse(dataset_df$group==1,0))


#Quantile normalization

normdataset=normalize.quantiles(t(as.matrix(dataset_df[,-ncol(dataset_df)])))


dataset_df[1:ncol(dataset_df)-1]=t(normdataset)


response <- 'group'

predictors <- setdiff(names(dataset_df), response)


performance_training=matrix( rep( 0, len=21), nrow = 3)

performance_testing=matrix( rep( 0, len=56), nrow = 8)
```

```
performance=matrix(rep( 0, len=56), nrow = 8)

performance_testing_list=list()
performance_training_list=list()

#Random sampling
rand <- sample(nrow(dataset_df))
dataset_df=dataset_df[rand, ]

#Randomly split the training dataset
trainIndex <- createDataPartition(dataset_df$group, p = .8,list = FALSE,times = 1)
irisTrain <- dataset_df[ trainIndex,]
irisTest <- dataset_df[-trainIndex,]
irisTrain$group=as.factor(paste0("X",irisTrain$group))
irisTest$group=as.factor(paste0("X",irisTest$group))

#10-fold crossvalidation
control <- trainControl(method="cv", number=10,classProbs = TRUE,summaryFunction =
twoClassSummary)

#DL algorithm
dataset.hex<-as.h2o(irisTrain, destination_frame="train.hex")
valid <- as.h2o(irisTest, destination_frame="test.hex")
```

```
dataset.hex$group <- as.factor(dataset.hex$group)
hyper_params <- list(
activation=c("Rectifier","Tanh","Maxout"),
hidden=list(c(20),c(40),c(60),c(20,20),c(40,40),c(60,60),c(20,20,20),c(40,40,40),c(60,60,60)),
input_dropout_ratio=c(0,0.05,0.1,0.2,0.3,0.4,0.5),
hidden_dropout_ratios=c(0.5,0.6),
l1=seq(0,1e-4,1e-5),
l2=seq(0,1e-4,1e-5),
train_samples_per_iteration =c(0,-1,-2),
epochs = c(1,5,10,20,30,40,50,60,70,80,90,100,200,300,400,500,600,700,800,900,1000),
momentum_start=c(0,0.5),
rho=c(0.9,0.95,0.99,0.995,0.999),
quantile_alpha=c(0,1),
huber_alpha=seq(0,1) )
m_loaded <- h2o.loadModel(path)
search_criteria = list(strategy = "RandomDiscrete", max_models = 100, stopping_rounds=5,
stopping_tolerance=1e-2)

training_frame=dataset.hex,
validation_frame=valid,
hidden=c(20,20),
x=predictors,
y="group",
```

```
seed=7,
epsilon=c(1e-4,1e-5,1e-6,1e-7,1e-8,1e-9,1e-10)
variable_importances=TRUE,
export_weights_and_biases=T,
standardize=T,
stopping_metric="misclassification",
stopping_tolerance=1e-2,
stopping_rounds=2,
score_validation_samples=10000,
score_duty_cycle=0.025,
max_w2=c(0,10),
hyper_params = hyper_params,
search_criteria = search_criteria,
nfolds=10
)
manual <- h2o.getmanual("dl_manual_randome1",sort_by="mse",decreasing=FALSE)
manual@summary_table[1,]
best_model <- h2o.getModel(manual@model_ids[[1]])

#Recall
performance_training[1,7]=as.numeric(best_model@model$cross_validation_metrics_summary
$mean)
#Precision
```

```
performance_training[2,7]=as.numeric(best_model@model$cross_validation_metrics_summary
$mean)
#F1 score
performance_training[3,7]=as.numeric(best_model@model$cross_validation_metrics_summary
$mean)
#Accuracy
performance_training[4,7]=as.numeric(best_model@model$cross_validation_metrics_summary
$mean)
#AUROC
performance_training[5,7]=as.numeric(best_model@model$cross_validation_metrics_summary
$mean)

perf=h2o.performance(best_model,as.h2o(irisTest, destination_frame="test.hex"))

#Recall
performance_testing[1,7]=as.numeric(h2o.sensitivity(perf,h2o.find_threshold_by_max_metric(p
erf,"f1"))[[1]])
#Precision
performance_testing[2,7]=as.numeric(h2o.precision(perf,h2o.find_threshold_by_max_metric(per
f,"f1")[[1]]))
#F1 score
performance_testing[3,7]=as.numeric(h2o.F1(perf,h2o.find_threshold_by_max_metric(perf,"f1")
)[[1]])
```

```
#Accuracy
performance_testing[4,7]=as.numeric(h2o.accuracy(perf,h2o.find_threshold_by_max_metric(per
f,"f1"))[[1]])
#AUROC
performance_testing[5,7]=as.numeric(h2o.auc(perf,h2o.find_threshold_by_max_metric(perf,"f1"
))[[1]])

performance_training_list[[k]]=performance_training
performance_testing_list[[k]]=performance_testing

#Performance of training dataset
performance_training=matrix( rep( 0, len=35), nrow = 5)

#Performance of test dataset
performance_testing=matrix( rep( 0, len=35), nrow = 5)
}
```

REFERENCES

1. Cook, D., Practical Machine Learning with H2O. *Powerful, Scaleable Technique for Deep Learning and AI*, 1st ed.; O'REILLY Media press: Sebastopol, CA, **2016**; pp 187-224.

2. LeCun, Y.; Bengio, Y.; Hinton, G., Deep learning. *Nature* **2015,** 521, (7553), 436-44.

SUPPORTING INFORMATION TABLES

Table S-1. Hyperparameter settings for optimization of deep learning method.

No.	epochs	nodes and hidden layers	activation function	rho	epsilon	L1, L2	input dropout ratio	hidden dropout ratio	training samples per iteration	max w2
1	1	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
2	5	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
3	10	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
4	20	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
5	30	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
6	40	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
7	50	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
8	60	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
9	70	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
10	80	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
11	90	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
12	100	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
13	200	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
14	300	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
15	400	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
16	500	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
17	600	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
18	700	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
19	800	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
20	900	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
21	1000	20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
22	400	20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
23	400	40	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
24	400	60	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
25	400	40-40	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
26	400	60-60	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
27	400	20-20-20	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
28	400	40-40-40	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10
29	400	60-60-60	rectifier	0.990	1.00E-08	0, 0	0	0.5	-2	10

30	400	20-20	tanh	0.990	1.00E-08	0, 0	0	0.5	-2	10
31	400	20-20	maxout	0.990	1.00E-08	0, 0	0	0.5	-2	10
32	400	20-20	rectifier	0.900	1.00E-08	0, 0	0	0.5	-2	10
33	400	20-20	rectifier	0.950	1.00E-08	0, 0	0	0.5	-2	10
34	400	20-20	rectifier	0.995	1.00E-08	0, 0	0	0.5	-2	10
35	400	20-20	rectifier	0.999	1.00E-08	0, 0	0	0.5	-2	10
36	400	20-20	rectifier	0.990	1.00E-04	0, 0	0	0.5	-2	10
37	400	20-20	rectifier	0.990	1.00E-05	0, 0	0	0.5	-2	10
38	400	20-20	rectifier	0.990	1.00E-06	0, 0	0	0.5	-2	10
39	400	20-20	rectifier	0.990	1.00E-07	0, 0	0	0.5	-2	10
40	400	20-20	rectifier	0.990	1.00E-09	0, 0	0	0.5	-2	10
41	400	20-20	rectifier	0.990	1.00E-10	0, 0	0	0.5	-2	10
42	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-4	0	0.5	-2	10
43	***400***	***20-20***	***rectifier***	***0.990***	***1.00E-08***	***0, 1.0E-5***	***0***	***0.5***	***-2***	***10***
44	400	20-20	rectifier	0.990	1.00E-08	1.0E-4, 0	0	0.5	-2	10
45	400	20-20	rectifier	0.990	1.00E-08	1.0E-4, 1.0E-4	0	0.5	-2	10
46	400	20-20	rectifier	0.990	1.00E-08	1.0E-4, 1.0E-5	0	0.5	-2	10
47	400	20-20	rectifier	0.990	1.00E-08	1.0E-5, 0	0	0.5	-2	10
48	400	20-20	rectifier	0.990	1.00E-08	1.0E-5, 1.0E-4	0	0.5	-2	10
49	400	20-20	rectifier	0.990	1.00E-08	1.0E-5, 1.0E-5	0	0.5	-2	10
50	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0.05	0.5	-2	10
51	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0.10	0.5	-2	10
52	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0.20	0.5	-2	10
53	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0.30	0.5	-2	10
54	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0.40	0.5	-2	10
55	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0.50	0.5	-2	10
56	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0	0.6	-2	10
57	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0	0.5	-1	10
58	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0	0.5	0	10
59	400	20-20	rectifier	0.990	1.00E-08	0, 1.0E-5	0	0.5	-2	0

Bold italic font denotes the conditions of the hyperparameter with best performance in deep learning method.

Table S-2. Performance of deep learning method according to hyperparameter settings.

No.	execution time (h:m:s)	number of optimized markers	Training data set					Test data set				
			Recall	Precision	F_1 score	accuracy	AUROC	Recall	Precision	F_1 score	accuracy	AUROC
1	00:02:00	1	0.9697	0.5246	0.6809	0.5652	0.4882	0.8481	0.3268	0.4718	0.5268	0.6349
2	00:05:10	6	0.7273	0.7742	0.7500	0.7681	0.8519	0.7975	0.6364	0.7079	0.8360	0.8985
3	00:09:19	11	0.7879	0.9286	0.8525	0.8696	0.9369	0.8734	0.5656	0.6866	0.8013	0.8765
4	00:13:00	13	0.6667	1.0000	0.8000	0.8406	0.9318	0.8481	0.6381	0.7283	0.8423	0.9028
5	00:18:22	16	0.7273	1.0000	0.8421	0.8696	0.9015	0.8481	0.6036	0.7053	0.8233	0.8987
6	00:12:18	8	0.6364	0.9545	0.7636	0.8116	0.8796	0.8608	0.5812	0.6939	0.8107	0.9081
7	00:18:50	13	0.6667	0.9565	0.7857	0.8261	0.8931	0.8228	0.5963	0.6915	0.8170	0.8914
8	00:22:21	12	0.7273	0.9231	0.8136	0.8406	0.9428	0.7848	0.6263	0.6966	0.8297	0.9185
9	00:18:07	9	0.6970	0.8214	0.7541	0.7826	0.8611	0.8734	0.6330	0.7340	0.8423	0.9084
10	00:24:02	13	0.7879	0.9286	0.8525	0.8696	0.9251	0.8861	0.6195	0.7292	0.8360	0.9179
11	00:23:43	12	0.6667	0.9565	0.7857	0.8261	0.8973	0.8608	0.5862	0.6974	0.8139	0.9061
12	00:25:45	12	0.7879	0.9630	0.8667	0.8841	0.9377	0.8734	0.6389	0.7380	0.8454	0.9277
13	00:41:46	12	0.6970	0.9200	0.7931	0.8261	0.9108	0.8608	0.6355	0.7312	0.8423	0.9220
14	00:49:17	10	0.8182	0.8710	0.8438	0.8551	0.9184	0.9114	0.6667	0.7701	0.8644	0.9319
15	01:34:45	17	0.8182	0.9000	0.8571	0.8696	0.9091	0.9367	0.6607	0.7749	0.8644	0.9453
16	01:21:46	11	0.6970	0.9583	0.8070	0.8406	0.8973	0.8734	0.6273	0.7302	0.8391	0.9217
17	01:23:45	9	0.6970	0.8846	0.7797	0.8116	0.8822	0.8734	0.6571	0.7500	0.8549	0.9364
18	02:15:15	12	0.8182	0.8182	0.8182	0.8261	0.8586	0.9241	0.6636	0.7725	0.8644	0.9181
19	02:03:41	11	0.7273	0.9231	0.8136	0.8406	0.9150	0.8861	0.6364	0.7407	0.8454	0.9207
20	02:32:57	13	0.6970	0.9200	0.7931	0.8261	0.8939	0.8987	0.6961	0.7845	0.8770	0.9295
21	03:08:19	14	0.7576	0.8929	0.8197	0.8406	0.9141	0.9494	0.6466	0.7692	0.8580	0.9346
22	00:36:24	11	0.7879	0.8966	0.8387	0.8551	0.9192	0.8987	0.6636	0.7634	0.8612	0.9271
23	00:54:36	10	0.7879	0.9286	0.8525	0.8696	0.9133	0.9241	0.6822	0.7849	0.8738	0.9305
24	01:48:04	11	0.7879	0.8966	0.8387	0.8551	0.9099	0.9241	0.6460	0.7604	0.8549	0.9285
25	02:33:20	11	0.8485	0.9032	0.8750	0.8841	0.9310	0.8987	0.6762	0.7717	0.8675	0.9385
26	04:35:41	11	0.8182	0.8710	0.8438	0.8551	0.9133	0.8734	0.6635	0.7541	0.8580	0.9328
27	01:32:10	9	0.7879	0.9286	0.8525	0.8696	0.9116	0.8734	0.6900	0.7709	0.8707	0.9303
28	04:12:06	10	0.8182	0.8710	0.8438	0.8551	0.9082	0.8734	0.6699	0.7582	0.8612	0.9354
29	07:45:40	10	0.7576	0.8621	0.8065	0.8261	0.8990	0.8608	0.6800	0.7598	0.8644	0.9275
30	02:05:15	7	0.7273	0.9600	0.8276	0.8551	0.9082	0.8608	0.5913	0.7010	0.8170	0.9019

31	04:50:47	12	0.7576	0.8929	0.8197	0.8406	0.9133	0.8987	0.6017	0.7208	0.8265	0.9184
32	01:30:06	14	0.7576	0.9615	0.8475	0.8696	0.9470	0.8608	0.6415	0.7351	0.8454	0.9252
33	01:27:53	14	0.7879	0.8966	0.8387	0.8551	0.9360	0.9114	0.6429	0.7539	0.8517	0.9248
34	00:45:34	7	0.7879	0.8387	0.8125	0.8261	0.8813	0.9241	0.6697	0.7766	0.8675	0.9251
35	01:05:51	12	0.7273	0.8000	0.7619	0.7826	0.8771	0.9114	0.6486	0.7579	0.8549	0.9367
36	00:32:16	8	0.6061	0.8696	0.7143	0.7681	0.8721	0.7089	0.5895	0.6437	0.8044	0.8635
37	00:27:56	6	0.8182	0.7714	0.7941	0.7971	0.8830	0.8228	0.5856	0.6842	0.8107	0.8815
38	00:44:59	9	0.7576	0.6944	0.7246	0.7246	0.8140	0.8861	0.6422	0.7447	0.8486	0.8941
39	00:51:21	9	0.7576	0.8621	0.8065	0.8261	0.8906	0.8608	0.6071	0.7120	0.8265	0.9204
40	01:42:05	15	0.6970	0.9200	0.7931	0.8261	0.9099	0.8861	0.6667	0.7609	0.8612	0.9305
41	01:37:20	13	0.6364	0.9545	0.7636	0.8116	0.9268	0.8861	0.6250	0.7330	0.8391	0.9240
42	01:13:15	11	0.7879	0.9286	0.8525	0.8696	0.9066	0.8987	0.6574	0.7594	0.8580	0.9292
43	***01:25:05***	***14***	***0.8182***	***0.9643***	***0.8852***	***0.8986***	***0.9301***	***0.9114***	***0.6923***	***0.7869***	***0.8770***	***0.9472***
44	00:59:23	9	0.7879	0.8667	0.8254	0.8406	0.8990	0.8481	0.6204	0.7166	0.8328	0.9187
45	01:01:11	9	0.7273	0.9600	0.8276	0.8551	0.8981	0.8481	0.6907	0.7614	0.8675	0.9351
46	01:24:19	14	0.8788	0.9063	0.8923	0.8986	0.9200	0.8987	0.6574	0.7594	0.8580	0.9420
47	00:58:07	9	0.7576	0.7576	0.7576	0.7681	0.9015	0.8354	0.6875	0.7543	0.8644	0.9283
48	01:02:59	9	0.6970	0.8846	0.7797	0.8116	0.8864	0.8987	0.6698	0.7676	0.8644	0.9226
49	01:04:35	10	0.7879	0.8387	0.8125	0.8261	0.8965	0.8987	0.6698	0.7676	0.8644	0.9325
50	01:15:00	12	0.7576	0.8929	0.8197	0.8406	0.9024	0.9114	0.6154	0.7347	0.8360	0.9213
51	00:58:40	9	0.6364	0.8750	0.7368	0.7826	0.8855	0.8734	0.6571	0.7500	0.8549	0.9236
52	01:01:55	9	0.7879	0.8966	0.8387	0.8551	0.8788	0.8987	0.5772	0.7030	0.8107	0.8926
53	01:18:53	9	0.8485	0.8235	0.8358	0.8406	0.8948	0.8101	0.6214	0.7033	0.8297	0.9101
54	01:06:17	10	0.7879	0.8667	0.8254	0.8406	0.9167	0.8354	0.5841	0.6875	0.8107	0.8829
55	00:54:07	8	0.7576	0.8621	0.8065	0.8261	0.8923	0.8734	0.6106	0.7188	0.8297	0.8797
56	01:10:17	11	0.7273	0.9600	0.8276	0.8551	0.9024	0.8608	0.6733	0.7556	0.8612	0.9418
57	01:08:23	9	0.7879	0.8387	0.8125	0.8261	0.8948	0.9241	0.6577	0.7684	0.8612	0.9288
58	01:19:28	11	0.7273	0.8571	0.7869	0.8116	0.8880	0.9241	0.6759	0.7807	0.8707	0.9390
59	01:11:25	11	0.7879	0.9286	0.8525	0.8696	0.9108	0.9620	0.6609	0.7835	0.8675	0.9467

Bold italic font denotes the conditions of the hyperparameter with the best performance in the deep learning method.

Table S-3. Comparison of performance between deep leaning method and five machine learning methods by bootstrapping validation.

Machine Learning Method	Execution time (h:m:s)	Number of optimized features (*N*)	Detailed information of optimal model	Training dataset									Test dataset							
				TN (*N*)	FN (*N*)	FP (*N*)	TP (*N*)	Recall	Precision	F_1 score	accuracy	AUROC	TN (*N*)	FN (*N*)	FP (*N*)	TP (*N*)	Recall	Precision	F_1 score	accuracy	AUROC
Logistic Regression (LR)	**03:50:35**	**10**	Coefficient of feature and intercept: TTHY.AADDTWEPFASGK = -0.847 ITIH4.AGFSWIEVTFK = -0.337 CLU.ASSIIDELFQDR = -0.749 LRG1.DLLLPQPDLR = 0.283 KLKB1.DSVTGTLPK = -4.225 C1R.DYFIATCK = 3.118 MBL2.FQASVATPR = 1.218 IGFBP2.LIQGAPTIR = 4.861 C5.NADYSYSVWK = 0.563 PROS1.NNLELSTPLK = 1.181 Intercept = -0.670	13120	2658	1754	10416	0.8333	0.8537	0.8434	0.8612	0.9252	207	16	31	63	0.7975	0.6702	0.7283	0.8517	0.9159
Naïve Bayes (NB)	**01:45:20**	**12**	Mean parameter of control vs case: TTHY.AADDTWEPFASGK = 2.844 vs 1.889 IGFBP3. ALAQCAPPPAVCAELVR = 0.545 vs 0.470 CLU.ASSIIDELFQDR = 2.163 vs 1.950 SEPP1.CINQLLCK = 0.940 vs 0.820 LRG1.DLLLPQPDLR = 1.214 vs 2.452 KLKB1.DSVTGTLPK = 0.817 vs 0.741 C1R.DYFIATCK = 0.699 vs 0.856 MBL2.FQASVATPR = 0.280 vs 0.516 IGFBP2.LIQGAPTIR = 0.147 vs 0.273 C5.NADYSYSVWK = 5.099 vs 7.468 CLU.TLLSNLEEAK = 2.513 vs 2.341 SERPINC1.VAEGTQVLELPFK = 2.351 vs 2.304	13513	4405	1381	8727	0.6581	0.8652	0.7476	0.8143	0.8795	209	26	29	53	0.6709	0.6463	0.6584	0.8265	0.8686
k-Nearest Neighbor (k-NN)	**05:13:26**	**6**	[a]Configuration of model: ITIH4.AGFSWIEVTFK, IGFBP3.ALAQCAPPPAVCAELVR, LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK, ECM1.ELLALIQLER, IGFBP2.LIQGAPTIR	11877	3771	3153	9325	0.6929	0.7928	0.7395	0.7778	0.8650	187	19	51	60	0.7595	0.5405	0.6316	0.7792	0.8463

Support Vector Machine (SVM)	**06:04:11**	**14**	Configuration of (kernel) model: TTHY.AADDTWEPFASGK = -0.526 ITIH4.AGFSWIEVTFK = -0.319 CLU.ASSIIDELFQDR = -0.265 LRG1.DLLLPQPDLR = 0.405 KLKB1.DSVTGTLPK = -0.754 C1R.DYFIATCK = 0.390 APOC1.EWFSETFQK = -0.078 MBL2.FQASVATPR = 0.291 IGFBP2.LIQGAPTIR = 0.611 C5.NADYSYSVWK = 1.050 PROS1.NNLELSTPLK = 0.257 CLU.TLLSNLEEAK = -0.518 C1S.TNFDNDIALVR = 0.002 CFI.VFSLQWGEVK = 0.320	13528	2845	1520	10212	0.8333	0.8000	0.8163	0.8381	0.9080	204	19	34	60	0.7595	0.6383	0.6936	0.8328	0.90517
Random Forest (RF)	**02:29:35**	**8**	Weight of features: KLKB1.DSVTGTLPK = 0.060 MBL2.FQASVATPR = 0.131 CLU.TLLSNLEEAK = 0.081 IGFBP2.LIQGAPTIR = 0.091 C5.NADYSYSVWK = 0.131 LRG1.DLLLPQPDLR = 0.232 PROS1.NNLELSTPLK = 0.128 TTHY.AADDTWEPFASGK = 0.146	13150	4701	1890	8259	0.7143	0.8120	0.7600	0.7842	0.8684	216	36	22	43	0.5443	0.6615	0.5972	0.8170	0.87512
Deep Learning (DL)	**10:40:08**	**11**	[a]Configuration of model: TTHY.AADDTWEPFASGK, SEPP1.CINQLLCK, LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK, C1R.DYFIATCK, ECM1.ELLALIQLER, C5.GGSASTWLTAFALR, IPSP.GFQQLLQELNQPR, IGFBP2.LIQGAPTIR, C5.NADYSYSVWK, CLU.TLLSNLEEAK	12513	1986	2290	11130	0.7460	0.9038	0.8174	0.8500	0.9264	187	5	51	74	0.9367	0.5920	0.7255	0.8233	0.931337

TN, true negative; FN, false negative; FP, false positive; TP, true positive.

[a] Inner workings are difficult to understand and explain.

Table S-4. Comparison of performance between deep leaning method and five machine learning methods by number of features.

Limit number of generations	Machine Learning Method	Training dataset											Test dataset								
		Execution time (h:m:s)	Number of optimized features (N)	Detailed information of optimal model	TN (*N*)	FN (*N*)	FP (*N*)	TP (*N*)	Recall	Precision	F_1 score	accuracy	AUROC	TN (*N*)	FN (*N*)	FP (*N*)	TP (*N*)	Recall	Precision	F_1 score	accuracy	AUROC
3	**Logistic Regression (LR)**	< 00:02:00	3	Coefficient of feature and intercept: LRG1.DLLLPQPDLR = 1.445 KLKB1.DSVTGTLPK = -4.284 C5.NADYSYSVWK = 0.276 Intercept = -0.813	323	91	46	231	.6061	.8696	.7143	.7681	.8561	208	23	30	56	.7089	.6512	.6788	.8328	.8880
	Naïve Bayes (NB)	< 00:02:00	3	Mean parameter of control vs case: TTHY.AADDTWEPFASGK = 2.844 vs 1.889 LRG1.DLLLPQPDLR = 1.214 vs 2.452 IGFBP2.LIQGAPTIR = 0.147 vs 0.273	332	138	37	184	.4545	.8333	.5882	.6957	.7685	201	34	37	45	.5696	.5488	.5590	.7760	.7620
	k-Nearest Neighbor (k-NN)	< 00:02:00	3	[a]Configuration of model: ITIH4.AGFSWIEVTFK, LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK	295	88	74	234	.5152	.6800	.5862	.6522	.8375	182	25	56	54	.6835	.4909	.5714	.7445	.8553
	Support Vector Machine (SVM)	< 00:02:00	3	Configuration of (kernel) model: LRG1.DLLLPQPDLR = 1.661 KLKB1.DSVTGTLPK = -1.274 C5.NADYSYSVWK = 0.773	327	103	42	219	.5758	.9048	.7037	.7681	.8519	208	26	30	53	.6709	.6386	.6543	.8233	.8811
	Random Forest (RF)	< 00:02:00	3	Weight of features: IGFBP2.LIQGAPTIR = 0.242 C5.NADYSYSVWK = 0.327 TTHY.AADDTWEPFASGK = 0.431	312	121	57	201	.3939	.7647	.5200	.6522	.8274	196	28	42	51	.6456	.5484	.5930	.7792	.8199
	Deep Learning (DL)	00:37:11	3	[a]Configuration of model: LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK, C5.NADYSYSVWK	301	72	68	250	.7879	.8387	.8125	.8261	.8737	189	10	49	69	.8734	.5847	.7005	.8139	.8992
6	**Logistic Regression (LR)**	< 00:02:00	6	Coefficient of feature and intercept: TTHY.AADDTWEPFASGK = -0.624 CLU.ASSIIDELFQDR = -0.678 LRG1.DLLLPQPDLR = 0.597 KLKB1.DSVTGTLPK = -5.078 C1R.DYFIATCK = 3.734 C5.NADYSYSVWK = 0.513 Intercept = -0.208	327	76	42	246	.6061	.9091	.7273	.7826	.8914	204	17	34	62	.7848	.6458	.7086	.8391	.9044
	Naïve Bayes (NB)	< 00:02:00	6	Mean parameter of control vs case: TTHY.AADDTWEPFASGK = 2.844 vs 1.889 IGFBP3. ALAQCAPPPAVCAELVR = 0.545 vs 0.470 LRG1.DLLLPQPDLR = 1.214 vs 2.452 KLKB1.DSVTGTLPK = 0.817 vs 0.741 MBL2.FQASVATPR = 0.280 vs 0.516 IGFBP2.LIQGAPTIR = 0.147 vs 0.273	336	130	33	192	.4545	.8333	.5882	.6957	.8636	204	33	34	46	.5823	.5750	.5786	.7886	.8052

k-Nearest Neighbor (k-NN)	< 00:02:00	6	[a]Configuration of model: ITIH4.AGFSWIEVTFK, IGFBP3.ALAQCAPPPAVCAELVR, LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK, BTD.ILSGDPYCEK, and IGFBP2.LIQGAPTIR	317	89	52	233	.6667	.7586	.7097	.7391	.8552	193	19	45	60	.7595	.5714	.6522	.7981	.8882
Support Vector Machine (SVM)	< 00:02:00	6	Configuration of (kernel) model: TTHY.AADDTWEPFASGK = -0.824 ITIH4.AGFSWIEVTFK = -0.575 LRG1.DLLLPQPDLR = 0.923 KLKB1.DSVTGTLPK = -1.045 C1R.DYFIATCK = 0.808 C5.NADYSYSVWK = 1.246	328	83	41	239	.6061	.9524	.7407	.7971	.8872	209	22	29	57	.7215	.6628	.6909	.8391	.8997
Random Forest (RF)	< 00:02:00	6	Weight of features: LRG1.VAAGAFQGLR = 0.178 ECM1.ELLALIQLER = 0.251 CLU.TLLSNLEEAK = 0.100 IGFBP2.LIQGAPTIR = 0.103 C5.NADYSYSVWK = 0.169 TTHY.AADDTWEPFASGK = 0.199	327	131	42	191	.3939	.8125	.5306	.6667	.8047	204	34	34	45	.5696	.5696	.5696	.7855	.8162
Deep Learning (DL)	00:53:06	6	[a]Configuration of model: SEPP1.CINQLLCK, LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK, IPSP.GFQQLLQELNQPR, IGFBP2.LIQGAPTIR, C5.NADYSYSVWK	312	60	57	262	.7879	.8667	.8254	.8406	.8763	202	9	36	70	.8861	.6604	.7568	.8580	.9192
Logistic Regression (LR)	< 00:02:00	10[a]	Coefficient of feature and intercept: TTHY.AADDTWEPFASGK = -0.636 CLU.ASSIIDELFQDR = -0.732 LRG1.DLLLPQPDLR = -0.037 KLKB1.DSVTGTLPK = -3.923 C1R.DYFIATCK = 2.371 MBL2.FQASVATPR = 1.367 IGFBP2.LIQGAPTIR = 6.317 C5.NADYSYSVWK = 0.549 CLU.TLLSNLEEAK = -0.859 CFI.VFSLQWGEVK = 0.606 Intercept = -0.246	326	65	43	257	.6667	.9167	.7719	.8116	.9108	198	16	40	63	.7975	.6117	.6923	.8233	.9036

Naïve Bayes (NB)	< 00:02:00	10	Mean parameter of control vs case: TTHY.AADDTWEPFASGK = 2.844 vs 1.889 IGFBP3. ALAQCAPPPAVCAELVR = 0.545 vs 0.470 CLU.ASSIIDELFQDR = 2.163 vs 1.950 SEPP1.CINQLLCK = 0.940 vs 0.820 LRG1.DLLLPQPDLR = 1.214 vs 2.452 KLKB1.DSVTGTLPK = 0.817 vs 0.741 MBL2.FQASVATPR = 0.280 vs 0.516 IGFBP2.LIQGAPTIR = 0.147 vs 0.273 C5.NADYSYSVWK = 5.099 vs 7.468 SERPINC1.VAEGTQVLELPFK = 2.351 vs 2.304	338	117	31	205	.5455	.9000	.6792	.7536	.8830	211	29	27	50	.6329	.6494	.6410	.8233	.8684
Support Vector Machine (SVM)	< 00:02:00	10	Configuration of (kernel) model: TTHY.AADDTWEPFASGK = -0.698 ITIH4.AGFSWIEVTFK = -0.443 LRG1.DLLLPQPDLR = 0.605 KLKB1.DSVTGTLPK = -0.688 C1R.DYFIATCK = 0.548 MBL2.FQASVATPR = 0.314 IGFBP2.LIQGAPTIR = 0.505 C5.NADYSYSVWK = 1.074 PROS1.NNLELSTPLK = 0.239 CLU.TLLSNLEEAK = -0.476	337	71	32	251	.6667	.9565	.7857	.8261	.8855	204	21	34	58	.7342	.6304	.6784	.8265	.8937
Deep Learning (DL)	01:14:46	10	[a]Configuration of model: TTHY.AADDTWEPFASGK, SEPP1.CINQLLCK, HRG.DGYLFQLLR, LRG1.DLLLPQPDLR, KLKB1.DSVTGTLPK, C1R.DYFIATCK, ECM1.ELLALIQLER, IPSP.GFQQLLQELNQPR, IGFBP2.LIQGAPTIR, C5.NADYSYSVWK	308	44	61	278	.6667	.8148	.7333	.7681	.8535	192	5	46	74	.9367	.6167	.7437	.8391	.9231

TN, true negative; FN, false negative; FP, false positive; TP, true positive.

[a] Inner workings are difficult to understand and explain.

[b] The kNN and RF algorithms cannot create an optimal model with 10 features, because both have 6 features for the optimal model.

SUPPORTING INFORMATION FIGURES

Figure S-1. Optimization of hyperparameters that were barely affected by the performance of deep learning.

The hyperparameters of deep learning were optimized, and the performance was evaluated based on the AUROC value in the independent test dataset (the x-axis corresponds to parameter values, and the y-axis refers to AUROC values). The number of nodes and hidden layers (A), rho (B), L1 & L2 regularization (C), hidden dropout ratio (D), training samples per iteration (E), and max w2 (F) were optimized.

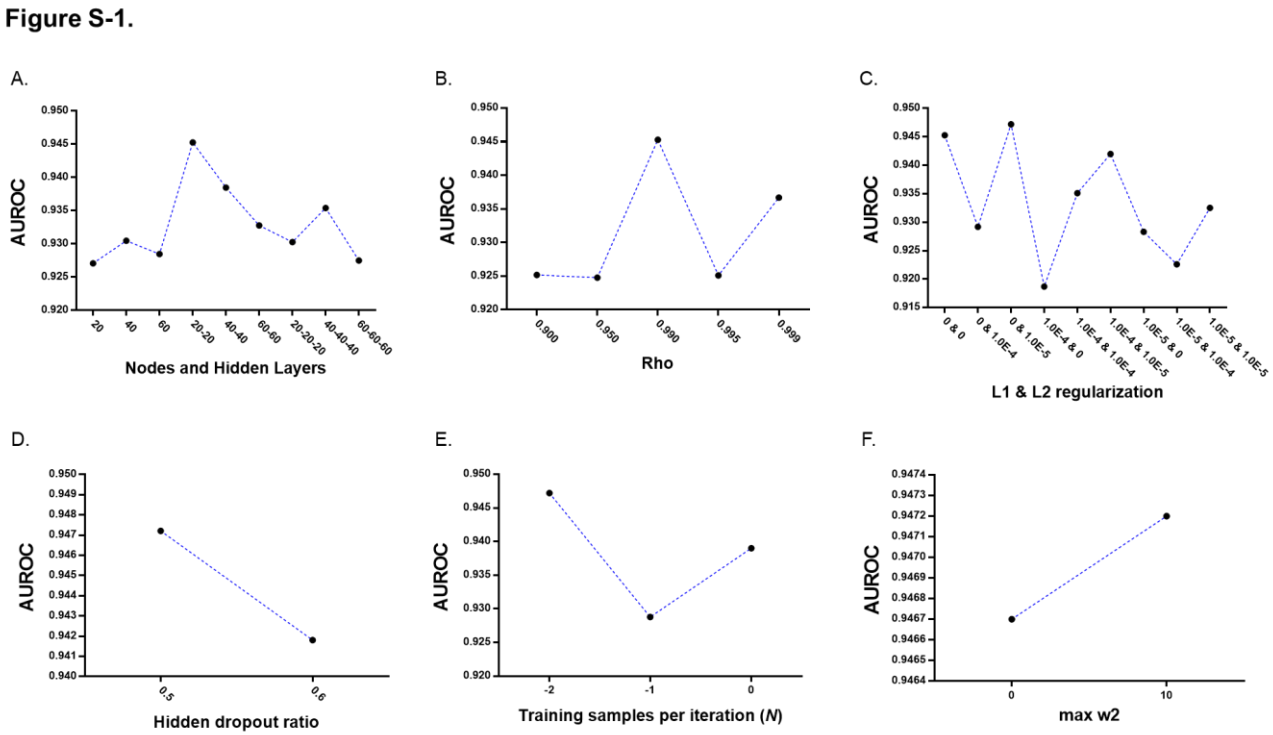

Figure S-2. Execution time according to optimization of 3 hyperparameters of the deep learning method.

Execution time according to epoch (A), nodes and hidden layers (B), and activations (C). As expected, as the number of epochs, nodes and hidden layers increased, the execution time became longer: rectifier was the fastest among activation functions.

Figure S-2.