# Deep-insight visible neural network (DI-VNN) for improving interpretability of a non-image deep learning model by data-driven ontology

**Herdiantri Sufriyana**

(1) Graduate Institute of Biomedical Informatics, College of Medical Science and Technology, Taipei Medical University, 250 Wu-Xing Street, Taipei 11031, Taiwan. (2) Department of Medical Physiology, Faculty of Medicine, Universitas Nahdlatul Ulama Surabaya, 57 Raya Jemursari Road, Surabaya 60237, Indonesia. https://orcid.org/0000-0001-9178-0222

**Yu Wei Wu**

(1) Graduate Institute of Biomedical Informatics, College of Medical Science and Technology, Taipei Medical University, 250 Wu-Xing Street, Taipei 11031, Taiwan. (2) Clinical Big Data Research Center, Taipei Medical University Hospital, 250 Wu-Xing Street, Taipei 11031, Taiwan. https://orcid.org/0000-0002-5603-1194

**Emily Chia-Yu Su** ( emilysu@tmu.edu.tw )

(1) Graduate Institute of Biomedical Informatics, College of Medical Science and Technology, Taipei Medical University, 250 Wu-Xing Street, Taipei 11031, Taiwan. (2) Clinical Big Data Research Center, Taipei Medical University Hospital, 250 Wu-Xing Street, Taipei 11031, Taiwan. (3) Research Center for Artificial Intelligence in Medicine, Taipei Medical University, 250 Wu-Xing Street, Taipei 11031, Taiwan. https://orcid.org/0000-0003-4801-5159

---

**Method Article**

**Keywords:** clinical prediction, deep learning, convolutional neural network, ontology, electronic health record

**Posted Date:** October 13th, 2021

**DOI:** https://doi.org/10.21203/rs.3.pex-1637/v1



---

# Abstract

We aimed to provide a framework that organizes internal properties of a convolutional neural network (CNN) model using non-image data to be interpretable by human. The interface was represented as ontology map and network respectively by dimensional reduction and hierarchical clustering techniques. The applicability is to implement a prediction model either to classify categorical or to estimate numerical outcome, including but not limited to that using data from electronic health records. This pipeline harnesses invention of CNN algorithms for non-image data while improving the depth of interpretability by data-driven ontology. However, the DI-VNN is only for exploration beyond its predictive ability, which requires further explanatory studies, and needs a human user with specific competences in medicine, statistics, and machine learning to explore the DI-VNN with high confidence. The key stages consisted of data preprocessing, differential analysis, feature mapping, network architecture construction, model training and validation, and exploratory analysis.


# Introduction

Clinical prediction in medicine have been extended by more machine learning algorithms with promising predictive performances, either to classify categorical or to estimate numerical outcomes.[1-4] Furthermore, recent deep-learning models were also be the most of well-known examples of prediction by machine, which surpasses human-level performance; yet, these were mostly for diagnosis.[5-7] Prognosis with causal reasoning, however, is more compelling. This is because clinical prediction is also expected to be actionable for prevention purpose which were mandated by growing implementation of insurance-based healthcare worldwide. While precision medicine and causal reasoning are very challenging problems in machine learning,[8] improving interpretability of a machine learning model can be an achievable solution to approach the goal in the near future.

To provide such solution, we have developed a deep-insight visible neural network (DI-VNN) as a human-machine interface for prediction tasks. The machine does what it is currently good at, which is handling large and complex data, and we organized multiple methods in an analysis pipeline such that the complexity could be understood by human users chunk-by-chunk, although this does not mean that it is easy to understand. Meanwhile, humans still do some things better than any machine, which is reasoning using extra data knowledge. To utilize both human and machine capabilities, we developed DI-VNN pipeline according to recent studies.[9,10] The difference is to construct a data-driven, convolutional neural network (CNN) architecture using non-image data with feature selection by differential analysis that deals high-dimensionality problem. The motivation was to achieve a moderate but acceptable predictive performance while allowing a human user to explore interactions among features in the context of predicting an outcome. Therefore, this model is expected to enable exploratory analyses along with external contextual knowledge.

Using this approach, our protocol allows a human to gain insights and to identify bias exploited for predictions by deeply exploring the ‘subconscious mind’ of a machine learning prediction model. Deep

exploration distinguished our pipeline compared to other interpretable clinical prediction. This means interpretation was facilitated by visual representations in an ontology connecting important features, semantically. For clinical implications, our models allow collaboration between humans and machines. Instead of developing the complete automation of clinical decision-making, a human-machine framework is more sensible for precision medicine. This framework also answers key challenges to evaluate model weaknesses,[11] verify if the predicted outcome is reasonable,[12] and make sensible clinical predictions by utilizing electronic medical records.[13] Eventually, DI-VNN also enables a human user to critically appraise and explore a prediction result case-by-case using extra data knowledge. This is similar to a common, long-standing practice for a doctor to learn and interpret clinical data for assessment of a patient's condition.

We applied this pipeline to several studies which were parts of a DI-VNN project. These studies applied our DI-VNN algorithm to a variety of predicted outcomes with comparison to those applying human learning with statistical methods and predictive modeling with other machine learning algorithms. Ethical clearance was waived by the Taipei Medical University Joint Institutional Review Board (TMU-JIRB number: N202106025). This protocol aimed to provide a framework that organizes internal properties of a CNN model using non-image data to be interpretable by human, which would be implemented as a prediction model, including but not limited to that using data from electronic health records. A 5-minute video is also available which explains the DI-VNN pipeline by focusing on the technical details (Supplementary Files).

# Reagents

# Equipment

To conduct most steps of the data analysis, we used R 4.0.2 programming language (R Foundation, Vienna, Austria). But, we also used Python 3.6.3 programming language (Anaconda Inc., Austin, TX, USA) for steps related to the DI-VNN. All codes for analysis were implemented using RStudio 1.3.959 (RStudio PBC, Boston, MA, USA). Bioconductor 3.11 was used to ensure reproducibility by synchronizing versions of the included R packages according to those in this Bioconductor version.[14] An R package of caret 6.0.86 was used for data partition. For the DI-VNN, we used keras 2.3.0 and tensorflow 2.0.0 Python libraries via R packages of reticulate 1.16, keras 2.3.0.0, and tensorflow 2.0.0. We also created R packages and a Python library for many steps in the data analysis, including the DI-VNN, medhist 0.1.0, clixo 0.1.1, and divnn 0.1.3 (both an R package and Python library). All of these packages/libraries are available for download from this repository https://github.com/herdiantrisufriyana. Details on other R package versions and all of the source codes (vignette) for the data analysis are available in https://github.com/herdiantrisufriyana/divnn_proc.

A set of hardware requirements may be needed to reproduce our work. We used a single machine for most of the data analysis, except training of the DI-VNN, with 16 logical processors for the 2.10 GHz

central processing unit (CPU) (Xeon® E5-2620 v4, Intel®, Santa Clara, CA, USA), 128 GB RAM, and 11 GB graphics processing unit (GPU) memory (GeForce GTX 1080 Ti, NVIDIA, Santa Clara, CA, USA). Meanwhile, the DI-VNN required a higher GPU capability than that provided by the previous specification. For hyperparameter tuning and training, we used multiple machines in a cloud with 90 GB RAM and 32 GB GPU memory (Tesla V100-SXM2, NVIDIA, Santa Clara, CA, USA). For predictions, the DI-VNN only needed a CPU in a local machine, or that in a cloud machine for the web application.

# Procedure

### 1. Split a dataset randomly for a training and validation set

We only used the training set to select features, create an ontology map and network, hyperparameter tuning, and training the model. It was blinded to the data distribution of any external validation sets. In this example, we developed a prediction model for prelabor rupture of membranes (PROM) using medical histories in electronic health records. If one also separates data for calibrating the model, then pre-calibration training set is suggested to be used solely for all of the procedures of the model development, as applied in our example.

### 2. Conduct quantile-to-quantile normalization

First, we applied quantile-to-quantile normalization. This means values from a sample are made to be similar to other samples quantile-to-quantile. We used the post-normalization, feature-wise average based on nationwide training set as the target of which quantile-to-quantile normalization of any subsets were applied onto. These included subsets for calibration and external validation. Instead of nationwide, these subsets were provider-wise, because we used these for prediction that likely uses medical history from a healthcare provider visited by a subject.

### 3. Select features by differential analysis

We conducted feature selection using a filter method that utilized a differential analysis. This analysis is commonly used in genomics, including microarray, RNA sequencing, and microbiome analyses.[15] In such fields, analyses deal with high-dimensional data with extremely low sample sizes relative to those of other biomedical fields. In this example, medical histories were the candidate features, either as individual or as multiple codes of diagnosis and procedure. We used nationwide, pre-calibration training set to conduct feature selection by differential analysis. Nationwide, instead of provider-wise, training set was used because we need to estimate the differential effect at population level. A differential effect means that a selected feature does not necessarily have the expected effect; however, the effect is more than the unselected one. Moderated *t*-statistics were applied, that is, fitting a univariable logistic regression for

each pair of candidate feature and outcome. Only a candidate feature with a significant differential effect was filtered into the downstream analysis. Since selection involves multiple regressions that are exposed to multiple testing effects, a procedure was applied for each significance using the Benjamini-Hochberg correction. This penalizes *p* values along with an increasing number of tests. A false discovery rate (FDR) or an adjusted *p* value of 0.05 was chosen as the lowest value for significance to exclude candidate predictors.

### 4. Apply 1-bit stochastic gradient descent transformation on the selected features

Using the average value of each feature from nationwide training set, we transformed normalized values into 0 if they were exactly the same as the average, 1 if they were greater than the average, and -1 otherwise. This is a 1-bit stochastic gradient descent transformation method which simplifies computations, thus allowing the training of deeper neural networks.[16] The biological relevance of this transformation is the same as undifferentiated, and up- and down-regulation. Under high-dimensional data perspectives, there is a chance that an increasing/decreasing factor is followed by changes in other factors to maintain values of particular ones, such that physiological homeostasis is maintained.

### 5. Compute correlation matrix using unnormalized candidate features for inferring feature map and network architecture

Demanding different statistical assumption, we used unnormalized candidate features for creating a feature map (i.e. ontology map) and a network architecture (i.e. ontology network). Both procedures need a distance/similarity matrix. We also used nationwide, pre-calibration training set to construct this matrix. Only the candidate features selected by differential analysis were used. Standardization was applied by subtracting each value with feature-wise average and dividing it with feature-wise standard deviation. Then, we computed a feature-to-feature Pearson correlation matrix.

### 6. Apply dimensional reduction to the filtered candidate features

For creating a feature map, we projected the filtered candidate features onto three dimensions. We called this dimensional reduction. To conduct this procedure, a dimensional reduction algorithm was applied, i.e. *t*-moderated stochastic network embedding (*t*-SNE) using the Barnes-Hut approximation.[17] To apply the Barnes-Hut *t*-SNE, an R package of Rtsne 0.15 is available to be downloaded from Bioconductor. Positioning of candidate features on the feature map was derived from pre-calibration training set. This positioning is also used for constructing the three-dimensional array of the other subsets in calibration and external validation. Applying of a dimensional reduction algorithm to create a feature map is a method called DeepInsight by the inventor.[9] The feature map is a multidimensional array (typically two

dimensions) which is the same input as for a CNN. It is a deep learning algorithm for extracting important features of an image, e.g., x-ray images. These features are fed-forward to a fully connected neural network model to predict an outcome. The CNN is the most successful deep learning algorithm in recent years.[18] Since it was developed for image data, its application to non-image data is conventionally limited. DeepInsight allows non-image data to be fed into the CNN model; therefore, we expected a state-of-the-art predictive performance from the constructed model. Although other methods were also available for the same purpose, the feature map of DeepInsight is data-driven. Dimensional reduction principally works to map features on a higher-dimensional space to a lower-dimensional space while minimizing information loss. This means two-dimensional output is optimized to represent many features and infer inter-feature distances. With real-world data, bivariate interactions may or may not occur only between a feature and an outcome, but inter-feature interactions are also other dynamics that may indirectly affect an outcome. In addition to the original values of features that reside on the feature map, distances between all possible pairs of features enlarge the hypothesis space for a CNN algorithm to fit representative weights to as many as variations available to predict an outcome.

### 7. Reduce the size of a feature map generated by dimensional reduction

The DI-VNN applied *t*-SNE in DeepInsight based on the rank or order instead of values of the *t*-SNE dimensions. If we have 5 candidate features, then we order these for axes of *x* and *y*. For example, a candidate feature is mapped at *x=1* and *y=3* in a two-dimensional space. This is because that feature has values on dimensions of *x* and *y*, that rank respectively at 1$^{st}$ and 3$^{rd}$ positions among the five candidate features. Then, we split axes of *x* and *y* into *7×7* grid; thus, there might be more than one candidate features at the same position. We rotated the map by convex-hull algorithm, as applied in the original DeepInsight, to find smallest feature map size then re-ranking the dimensions. For any positions, we computed maximum number of candidate features residing on the same position. That number determined the number of the two-dimensional layers, i.e. channels. The overlapped candidate features were ranked based on the third dimension. The ranks determined which channel a candidate feature residing on for each position in the two-dimensional space. Ranking the dimension is a novel method in the DI-VNN to reduce array dimensions, different to that applied by the previous DeepInsight transformation. This approach greatly reduced the feature map size; thus, a lighter array size was achieved when training the DI-VNN model.

### 8. Create an ontology array based on the feature map

Feature map in DI-VNN can be considered as a three-dimensional array. In each position, a candidate feature may have a value of -1, 0, or 1, depending on the 1-bit stochastic gradient descent transformation. For any positions in the array, that have no candidate feature residing on, a value of 0 is applied. The empty position may be occupied by an unfiltered candidate feature had it surpassed filtering by the

differential analysis. Therefore, zero value for the empty position has the same notion with either the unfiltered candidate features or those with normalized values equal to the feature-wise averages, i.e. undifferentiated features in relative to the outcome.

### 9. Infer an ontology from the data for network architecture or ontology network

Using the same correlation matrix with that for feature mapping, we inferred an ontology from pre-calibration training set. To conduct this procedure, a hierarchical clustering algorithm was applied, i.e. clique-extracted ontology (CliXO).[19] Since CliXO is implemented in C++ programming language, we developed an R package of clixo 0.1.1 for simpler implementation of CliXO by R users. In our example, we applied hyperparameters $\alpha=0.01$ and $\beta=0.5$, as recommended by the inventor. Unlike DeepInsight, application of a hierarchical clustering algorithm to create a network architecture is motivated by a need to improve interpretability by introducing transparency of internal properties of a neural network. Originally called a visible neural network by the inventor,[10] this method constructs a data-driven neural network architecture. Specifically, this method applied the CliXO algorithm which can be considered agglomerative clustering similar to complete- or single-linkage hierarchical clustering. Unlike many hierarchical clustering algorithms, the CliXO algorithm respects biological pleiotropy, which is an entity that may belong to one or more entities. For example, in genomics, a child term of gene ontology (GO) for biological processes is a part of >1 parent terms. CliXO precisely mimics the human-curated GO.[19] With an ontology alignment algorithm, the CliXO may or may not be aligned to an existing GO, but one that is not aligned may be a novel ontology term. While no ontology database is known for medical histories for any diseases, we may expect semantical groupings among features. This means that we expect those within the same group to also be ones that have similar conditions in a particular circumstance.

### 10. Add root ontology to ensure all candidate features are included by any ontology

Since CliXO is an agglomerative hierarchical clustering algorithm, ontologies may unite into the highest ontology in the hierarchy. In this example, the unified ontology included all candidate features. But, this may not be the case. Anyway, an ontology, called root ontology, was added to include all candidate features, either from the remaining candidate features, if any, or ones in the ontology consisting the most candidate predictors (i.e. highest ontology in the hierarchy).

### 11. Subset an ontology array for each instance into different subsets based on the inferred ontology

The resulting ontologies derived from pre-calibration training set were used to subset an ontology array into multiple arrays, including those in calibration and external validation. If we have 5 candidate features, feature 1 and 5 may be clustered into the same ontology while feature 3 and 4 into another

ontology. To subset the ontology array into one for the first ontology, we multiplied all numbers in the array with 0 except the feature members of that ontology, which are feature 1 and 5. The same method is applied for the ontology including feature 3 and 4. Eventually, for each instance, we had an array for each ontology derived by CliXO algorithm.

### 12. Construct a CNN architecture based on the final ontology

The original visible neural network only uses the vanilla neural network instead of the CNN. This neural network model might not represent all variations within the data as we expect from a CNN. Meanwhile, DeepInsight allows application of the CNN. Therefore, we developed this pipeline that organizes DeepInsight and a visible neural network into the DI-VNN model. Unlike sequential CNNs, the DI-VNN isolates backpropagation following the hierarchical structure of CliXO. Each ontology array was fed to a block of neural network, i.e. Inception v4-Resnet. We applied this type of architecture because it enables faster and lighter computation;[20] thus, we did not need to simplify the network architecture by CliXO to obtain an acceptable speed for training. In this architecture, an array in terminal branch of an ontology hierarchy is filtered by feeding it through Inception v4, then reconstructing the array into the same dimensions as it entered the Resnet architecture. Element-wise addition is applied between the filtered array and the original one. This mathematical operation is the Resnet architecture itself. To connect a child ontology array with a sibling ontology, if any, we applied depth concatenation, i.e. by the channel. The concatenated array (double or more channels) is fed to the Inception v4, then reconstructing the array into the same dimensions with the parent ontology array. The same mathematical operation is applied for the concatenated, filtered array and the parent one. Child-to-parent connection was applied between one or more arrays in non-terminal branch of an ontology hierarchy and the parent array. Each of either the terminal or non-terminal branch had a transformed array as a representation of an input array of each ontology.

### 13. Assign random weights in the constructed CNN architecture

Unlike regression algorithms, a neural network begins with random weights. If the randomization is inappropriate, the updated weights or parameters may be too high or too low, thus preventing the optimal combination of weights to be achieved during the iterations. In our example, we applied the He and Glorot methods for initial randomization respectively to the hidden layer (between the deepest and most surficial layers) and the output (the most surficial) layer.[21,22]

### 14. Compile a CNN model including the architecture and loss function

We need to know how each ontology array was transformed into a single number from 0 to 1 that predicts a probability of an event, or an integer for estimation task. After transformation by a block of Inception v4-Resnet, the transformed array was fed to a block of layers for convolution. This reduced the dimensions from $7\times7$ in each channel into a single number by a series of mathematical operation of convolution. Therefore, we have multiple numbers showing probabilities of an event for each instance, transformed and convoluted from all ontology arrays. Later for each iteration of the model training, we computed differences between the outcome, which is 0 and 1 respectively for nonevent and event, and each probability convoluted from each ontology array. For estimation task, this was the differences between the true time and the predicted time of delivery. Loss function (equation in Figure 1) was applied with $\alpha=0$ and $\lambda=\{10^{-9},10^{-8},\cdots,10^{0}\}$ to compute errors that were minimized by the training. We computed errors for each ontology term or node, not only the root node that we used for prediction. The errors of non-root nodes, $t$, were weighted as much as $\gamma=0.3$ of that of the root node, $r$ (equation in Figure 2), as previously used in GoogLeNet[23] and the initial model of the visible neural network[10]. All losses were normalized to within 0 to 1.

## 15. Determine batch size to compute loss for each iteration

To get a learning representation from the data, we would train the model in a series of iterations using backpropagation algorithm. This is a sequential computation of a loss function from the surface layer, which is closer to the outcome (depicted on top), to a deeper layer, which is closer to the features (depicted on the bottom or leaf nodes). In the DI-VNN, the correction is made by considering similarities (represented as distances) among features layer-by-layer from the surface (more-common ontology terms) to deeper layers (more-specific ontology terms). Originally, a visible neural network maps genotypes in the deeper layer to phenotypes in the surface layer by ontotypes which are ontology terms convoluted from deep to surface layers. Thus, we expect to find specific insights in deeper layers in the context of predicting an outcome. In our example, a batch size of 512 instances was computed for the loss or error in each iteration. The loss was used to update the weights. This means an error of prediction is iteratively corrected to the lowest possible level given the maximum iterations and the smallest improvement to pursue, as chosen by a human user. The weights were initiated randomly when constructing the architecture, then the weights were updated via backpropagation from the root ontology to the terminal ones. This procedure is iterative using 512 instances each time until 80% instances were used in pre-calibration training set. This achieved a cycle of iterations, or epoch.

## 16. Determine learning rate to update weights for each iteration

At the end of each epoch, we computed the loss and AUROC using another 20% instances. Each update was multiplied by a number, called learning rate. We applied a learning rate of $2^{-6}$, but this would be systematically changed during the model training. From an epoch to the next one, the learning rate for the

next epoch was reduced by 4% if the AUROC of the 20%-validation subset was no higher than 0.01 in addition to that of the previous epoch. No reduction was applied if reaching the minimum, which is 1/512 of the initial learning rate.

### 17. Consider to use warm-up strategy to begin each epoch

In our example, for the iterations covering the first 5% of instances in each epoch, the learning rate was initiated at one thirty-second of the learning rate at the first iteration for that epoch. This is gradually increased until reaching the learning rate at the last iteration covering the first 5% of instances, i.e. warm-up strategy.[24] This allowed a larger batch size, for which a number of samples is used for updating parameters at each step of the iteration until all training samples are used in each epoch. It is a term for a complete cycle to update parameters using all samples. Commonly, a smaller batch size maximizes the predictive performance at the cost of a longer training time; thus, training the model using the warm-up strategy required a shorter time without sacrificing the predictive performance. A larger batch size also demands a lower learning rate. Moreover, the learning rate is reduced across epochs if no improvement is made in the predictive performance per epoch compared to that of the previous one.[20] A lower learning rate makes parameter updating subtler; thus, the parameters do not jump too much in pursuit of optimal values, allowing the predictive performance to be further improved compared to that without decreasing the learning rate. All of these empirical approaches were chosen to allow realistic computations in terms of time and capacity for most architectures constructed by CliXO and to achieve acceptable predictive performances.

### 18. Determine criteria to stop the model training

Maximum epochs of 5 and 500 was set respectively for the hyperparameter tuning and the final training. After 50% of the maximum epochs, the iterations were stopped earlier if the AUROC of the 20%-validation subset was no higher than 0.001 in addition to that of the previous epoch. Only weights of the best iteration were finally used. This was determined based on the validation AUROC.

### 19. Determine model evaluation technique after the training is finalized

In our example, after stopping, we computed AUROCs of the 20% validation subsets by bootstrapping for 30 times. Since this model is computationally expensive, we trained this model using ~80% of the pre-calibrated set and validated the predictive performance using the remaining ~20% for each epoch, as described in the previous steps.

### 20. Conduct hyperparameter tuning and finally train the model using the best hyperparameter

All procedures from the batch size to stopping criteria were applied for each trial in the hyperparameter tuning and the final training. The tuning grid was applying different λ values (equation in Figure 1). In our example, the best tuning parameter was determined by the bootstrapped, validation AUROC.

### 21. Consider to calibrate the model

In our example, the DI-VNN was calibrated using a general additive model using locally weighted scatterplot smoothing (GAM-LOESS). The calibration used a ~20% split of a training set (~64% of all selected visits). The calibration set may be used to compare the predictive performance with other models, if any.

### 22. Extract learning representations from the trained model for population-level exploration and interpretation

For a population-level exploration using the DI-VNN, we extracted arrays of intermediate outputs for all ontology terms, including the root one. The intermediate output was an array after being filtered by each block of Inception v4-Resnet but before being convoluted into a single number as a predicted probability inferred from the neural network for each ontology term. The intermediate array had the same dimensions as the input array. An array was extracted for each instance and ontology term. We computed element-wise average values for each ontology array across all instances with the same outcome, which could be an event or nonevent. For the estimation task, we computed those values per outcome, as if it was an event or nonevent, respectively, as: (1) less than or equal to the average time of delivery; or (2) greater than the average time of delivery. We then computed element-wise subtraction of the average array of the event with that of the nonevent for each ontology term. Since a feature member of an ontology term resided at a particular position in the array, the subtracted array showed values that might surround that of the feature member. This value might be positive or negative. However, positive and negative outputs were not straightforward respective to events and nonevents. Yet, these tended to contribute to opposite outcomes. Interpretation should apply extra data knowledge using results of causal inferences.

### 23. Extract a learning representation from the trained model for individual-level exploration and interpretation

We also visualized the ontology network and array for individual-level exploration. But, obviously, we did not compute the average value for the array. The intermediate outputs were shown for an ontology array

that was chosen by the user. Either positive or negative values at this individual level had the same meaning as those at the population level. The predicted outcomes were also shown based on parts of the architecture up to each node (ontology). A value for the AUROC was also shown for an ontology chosen by the user for the array. However, this metric was computed using the pre-calibrated DI-VNN only. This is because the calibrated DI-VNN used all parts of the architecture up to the root ontology array.

# Troubleshooting

### Step 1

*Problem*

Premature stop of computation

*Possible reason*

Dataset consists of a large sample size or number of variables.

*Solution*

Consider to use computer with larger memory size or RAM. Alternatively, split table into ones consisting unique sets of variables.

### Step 2 and 4

*Problem*

Missing values of all instances for a feature

*Possible reason*

There is missing values in any instance for a feature.

*Solution*

Apply imputation to those values.

### Step 3

*Problem*

No selected feature

*Possible reason*

The number of candidate features is too small.

*Solution*

Consider to increase maximum FDR to select a candidate feature, or include all candidate features.

**Step 5**

*Problem*

Premature stop of computation

*Possible reason*

Dataset consists of a large number of variables.

*Solution*

Consider to use computer with larger memory size or RAM. Alternatively, create several tables splitting irredundant pair of features, then compute the correlation coefficients batch-by-batch.

**Step 6 to 8**

*Problem*

Unexpected error

*Possible reason*

The number of candidate features is too small.

*Solution*

Decrease perplexity of *t*-SNE step-by step down to minimum of 2. Alternatively, consider to increase maximum FDR to select a candidate feature, or include all candidate features.

**Step 9 to 11**

*Problem*

Unable to infer ontology

*Possible reason*

The number of candidate features is too small.

*Solution*

Consider to increase maximum FDR to select a candidate feature, or include all candidate features. Alternatively, create pseudo-ontologies with one root and another ontology.

### Step 12 to 16 and 20

*Problem*

Premature stop of computation

*Possible reason*

Dataset consists of a large sample size or complex architecture.

*Solution*

Consider to use computer with larger GPU memory size. Alternatively, reduce batch size and adjust learning rate accordingly.

### Step 17

*Problem*

Absence of warm-up (within only 1 step)

*Possible reason*

The number of steps is too small

*Solution*

Reduce batch size and adjust learning rate accordingly.

### Step 18 to 19

*Problem*

Slow processing

*Possible reason*

Dataset consists of a large sample size or complex architecture.

*Solution*

Consider to use computer with larger GPU memory size. Alternatively, increase batch size and adjust learning rate accordingly, stop training earlier, or reduce number of bootstrapping.

**Step 21**

*Problem*

Poorer predicted performance after calibration

*Possible reason*

The range of predicted probability is too narrow.

*Solution*

Consider without calibration.

**Step 22 to 23**

*Problem*

Slow processing

*Possible reason*

Dataset consists of a large sample size or complex architecture.

*Solution*

Consider to use computer with larger GPU memory size.

# Time Taken

### All step

Approximate time: 2 to 22 hours (pre-computed)

### Step 1

Approximate time: 10 to 20 minutes (pre-computed)

### Step 2 to 13 and 19

Approximate time: 3 to 5 minutes

### Step 14 to 18 and 20

Approximate time: 1 to 20 hours per classification/estimation model

### Step 21

Approximate time: 10 to 20 minutes

### Step 22

Approximate time: 1 to 2 minutes

### Step 23

Approximate time: 10 to 15 minutes

## Anticipated Results

The DI-VNN was developed to achieve moderate predictive performance but interpretable results. We may expect calibrated DI-VNN having better calibration compared with the non-calibrated one, but this may be not the case. In either population- or individual level exploration, we may expect some ontology arrays to be visually distinguished; thus, we can identify important features and the connections among these features. A positive output does not necessarily refer to an event. Nevertheless, positive and negative

(color-coded) outputs tend to contribute to opposite outcomes, which are interpreted based on external contextual knowledge. Although the value of a feature member was zero, it may be important if it was next to higher values that supported an outcome. We can trace the nodes on the ontology and assess if the corresponding arrays have a similar value distribution on the same or adjacent dimensions.

Exploring this DI-VNN should be done with caution, since we developed the model using medical histories, which are diagnosis or procedure codes provided by medical doctors. We may or may not be modeling a human pathophysiology, but, we are definitely modeling the doctors' behaviors of coding a diagnosis or procedure.[25] By providing an interface to the internal properties of this model, a human user can assess each prediction case-by-case.

# References


1. Sufriyana, H., et al. Comparison of Multivariable Logistic Regression and Other Machine Learning Algorithms for Prognostic Prediction Studies in Pregnancy Care: Systematic Review and Meta-Analysis. JMIR Med Inform 8, e16503 (2020).

2. Fleuren, L.M., et al. Machine learning for the prediction of sepsis: a systematic review and meta-analysis of diagnostic test accuracy. Intensive Care Med 46, 383-400 (2020).

3. Lee, Y., et al. Applications of machine learning algorithms to predict therapeutic outcomes in depression: A meta-analysis and systematic review. J Affect Disord 241, 519-532 (2018).

4. Gonem, S., Janssens, W., Das, N. & Topalovic, M. Applications of artificial intelligence and machine learning in respiratory medicine. Thorax 75, 695-701 (2020).

5. Bien, N., et al. Deep-learning-assisted diagnosis for knee magnetic resonance imaging: Development and retrospective validation of MRNet. PLoS Med 15, e1002699 (2018).

6. Hannun, A.Y., et al. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nat Med 25, 65-69 (2019).

7. Rajpurkar, P., et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. PLoS Med 15, e1002686 (2018).

8. Wilkinson, J., et al. Time to reality check the promises of machine learning-powered precision medicine. Lancet Digit Health 2, e677-e680 (2020).

9. Sharma, A., Vans, E., Shigemizu, D., Boroevich, K.A. & Tsunoda, T. DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. Sci Rep 9, 11399 (2019).

10. Ma, J., et al. Using deep learning to model the hierarchical structure and function of a cell. Nat Methods 15, 290-298 (2018).

11. Yu, K.H., Beam, A.L. & Kohane, I.S. Artificial intelligence in healthcare. Nat Biomed Eng 2, 719-731 (2018).

12. Rajkomar, A., Dean, J. & Kohane, I. Machine Learning in Medicine. N Engl J Med 380, 1347-1358 (2019).

13. Beam, A.L. & Kohane, I.S. Big Data and Machine Learning in Health Care. Jama 319, 1317-1318 (2018).

14. Huber, W., et al. Orchestrating high-throughput genomic analysis with Bioconductor. Nat Methods 12, 115-121 (2015).

15. Ritchie, M.E., et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res 43, e47 (2015).

16. Seide, F., Fu, H., Droppo, J., Li, G. & Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. in INTERSPEECH (2014).

17. Maaten, L.V.D. Accelerating t-SNE using tree-based algorithms. J. Mach. Learn. Res. 15, 3221–3245 (2014).

18. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436-444 (2015).

19. Kramer, M., Dutkowski, J., Yu, M., Bafna, V. & Ideker, T. Inferring gene ontologies from pairwise similarity data. Bioinformatics 30, i34-42 (2014).

20. Filonenko, A., Kurnianggoro, L. & Jo, K. Comparative study of modern convolutional neural networks for smoke detection on image data. in 2017 10th International Conference on Human System Interactions (HSI) 64-68 (2017).

21. He, K., Zhang, X., Ren, S. & Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. in 2015 IEEE International Conference on Computer Vision (ICCV) 1026-1034 (2015).

22. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Vol. 9 (eds. Yee Whye, T. & Mike, T.) 249–256 (PMLR, Proceedings of Machine Learning Research, 2010).

23. Szegedy, C., et al. Going deeper with convolutions. in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1-9 (2015).

24. Goyal, P., et al. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. Preprint at https://arxiv.org/abs/1706.02677 (2017).

25. Beaulieu-Jones, B.K., et al. Machine learning for patient risk stratification: standing on, or looking over, the shoulders of clinicians? NPJ Digit Med 4, 62 (2021).

## Acknowledgements

The social security administrator for health or *badan penyelenggara jaminan sosial (BPJS) kesehatan* in Indonesia gave permission to access the sample dataset in this protocol (dataset request approval number: 5064/I.2/0421). This protocol was funded by the Ministry of Science and Technology (MOST) in Taiwan (grant number MOST109-2221-E-038-018 and MOST110-2628-E-038-001) and the Higher Education Sprout Project from the Ministry of Education (MOE) in Taiwan (grant number DP2-110-21121-01-A-13) to Emily Chia-Yu Su.

## Figures

$$L = \sum_{i=1}^{n} w_i \left( y_i - f(x_{ij}, \theta_j) \right)^2 + \alpha \sum_{j=0}^{p} \|\theta_j\| + \lambda \sum_{j=0}^{p} \|\theta_j\|^2$$

**Figure 1**

Equation of loss function with regularization

$$L = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{L_i^{(r)} + \gamma \sum_{t \neq r} L_i^{(t)}}{r + \gamma \times t} \right)$$

**Figure 2**

Equation of DI-VNN loss function

## Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- suppprotocoldivnn.pdf

- suppvideo.mov