

A APPENDIX

A.1 DLGN ILLUSTRATIONS

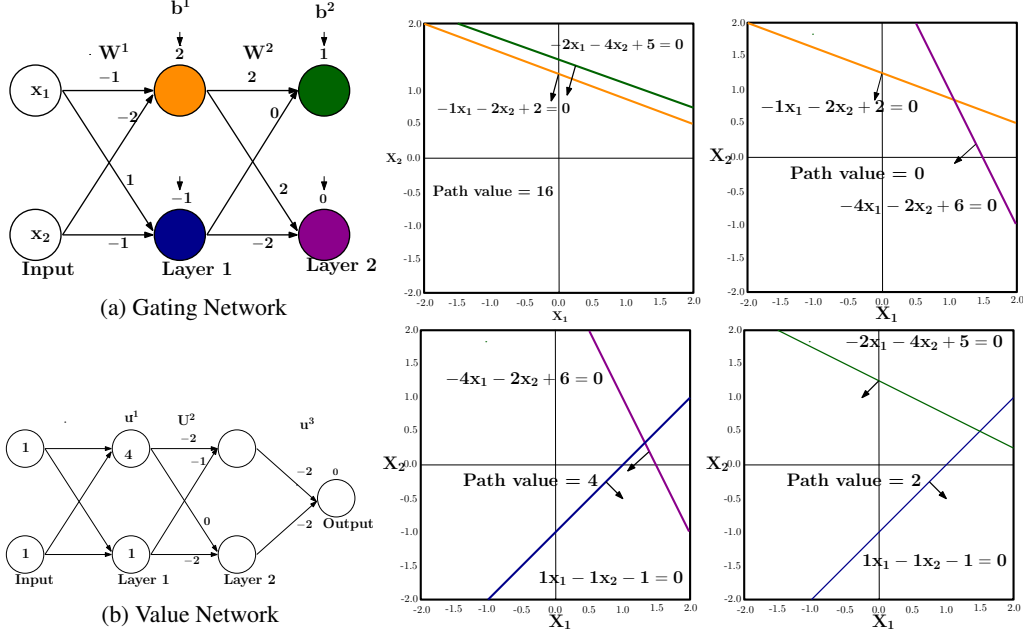


Figure 4: An illustration of DLGN network and node hyperplanes

Figure 4(a) and 4(b) give an example DLGN operating on a 2-dimensional input with 2 layers and 2 neurons per layer, i.e. $d = m = L = 2$. Figure 4(c) illustrates the region of activation for all the $m^L = 4$ paths given by the gating network 4(a). The value network in Figure 4(b) gives the path value for all these paths by multiplying the appropriate weights in the path.

Next, we will discuss various variants of DLGN.

A.2 DLGN-VT AND DLGN-SF

We also consider two natural variants of the DLGN architecture. The first is a simple re-parameterisation, where the gating network f_π is parameterised directly by the matrices V^1, V^2, \dots, V^L , each of shape $m \times d$, instead of using the parameters W^1, \dots, W^L . Clearly, this reparameterisation does not lose any representation power. We call this parameterisation as DLGN-SF (for shallow features).

Another variant that we use an explicit parameterisation of the coefficients g_π instead of using a value network. This parameterisation is also clearly more powerful than the standard DLGN parameterisation in the previous section. However, this requires a parameter tensor of size m^L and is not really practical for large L and m . We call this reparameterisation as DLGN-VT (for value tensor).

All of these models can be extended to allow a bias parameter for the gating network so that the functions η^ℓ can also be affine functions instead of strict linear functions, but we do not discuss this version here for the purpose of simplicity.

Number of DLGN hyperplanes (after training) within a given distance of the label function ODT hyperplanes of DLGN-VT and DLGN-SF are shown in Table 5 and Table 4. At initialization, all these numbers are equal to zero. The 15 ODT internal nodes are numbered 0 to 14, with 0 as the root. Results of similar experiments as DLGN (Table 1) shown in Table 4 and 5 for DLGN_SF and DLGN_VT respectively depicts identical hyperplane-seeking properties of these two variants

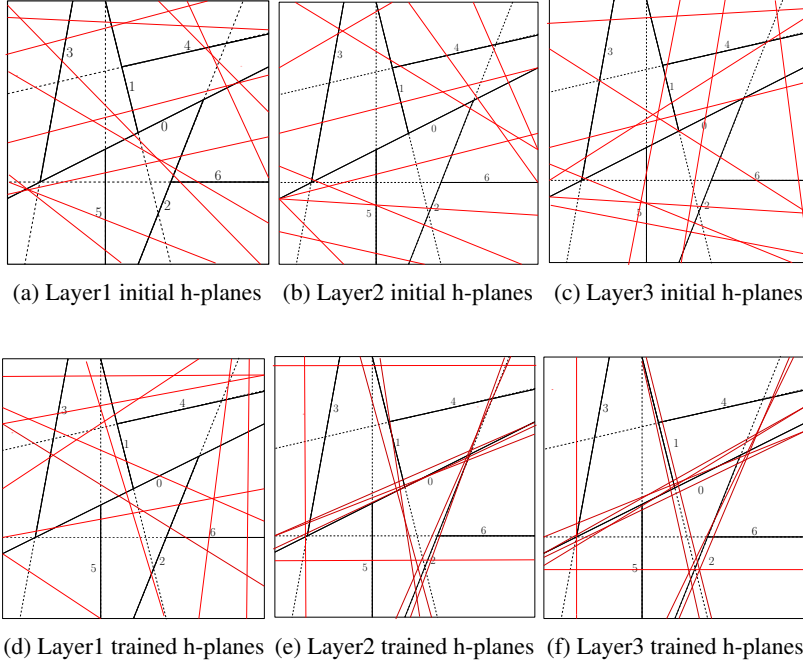


Figure 5: An illustration of DLGN hyperplanes before and after training on data in Fig 1c.

of DLGN models as well. For each node in the ODT, we count the number of DLGN hyperplanes within a distance of 0.1, 0.2, and 0.3 from it.

Table 4: Distance table for DLGN-SF Layers: 4 Nodes per layer: 10

Distance	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.1	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0
0.2	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0
0.3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Closest Dist.(init)	1.20	1.22	1.26	1.27	1.24	1.24	1.25	1.22	1.29	1.27	1.24	1.26	1.15	1.26	1.25
Closest Dist.(final)	0.05	0.07	0.08	0.08	0.12	0.11	0.13	0.09	0.18	0.26	0.08	0.16	0.10	0.14	0.21

Table 5: Distance table for DLGN-VT Layers: 4 Nodes per layer: 10

Distance	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.1	3	2	2	0	2	2	1	1	0	1	2	1	1	1	1
0.2	3	2	2	0	2	2	1	1	0	2	2	1	1	1	1
0.3	3	2	2	0	2	2	1	1	1	2	2	1	1	1	1
Closest Dist.(init)	1.16	1.27	1.23	1.22	1.28	1.22	1.18	1.17	1.27	1.26	1.19	1.21	1.26	1.23	1.21
Closest Dist.(final)	0.03	0.05	0.06	0.35	0.08	0.09	0.06	0.09	0.24	0.06	0.04	0.07	0.05	0.07	0.05

A.3 DETAILS OF SUBROUTINES USED IN DLGN-DT

This function takes into account the DLGN model after training. It iteratively calculates the effective weights of the layer using the formula $V^\ell = W^\ell W^{\ell-1} \dots W^1$ as in lines 3 of the algorithm. It finally returns each layer’s effective weight (hyperplanes) as a vector.

Algorithm 3 Return gates of a trained DLGN model**Arguments:** A DLGN model with parameters $W^1, \dots, W^L, \mathbf{u}^1, U^2, \dots, U^L, \mathbf{u}^{L+1}$ **Outputs:** mL hyperplanes in the input dimension

```

1: function GATEHYPERPLANES(model)
2:   for  $l \leftarrow 1$  to  $L$  do
3:      $V^l \leftarrow W^l W^{l-1} \dots W^1$ 
4:   end for
5:   return  $V \leftarrow V^1, \dots, V^L$ 
6: end function

```

A.4 DATASETS USED

We have used 3 synthetic datasets and 20 tabular datasets to evaluate the performance of our model DLGN and its variants against some standard algorithms. In this section, we will elaborately describe the datasets used.

A.4.1 SYNTHETIC DATASETS:

Table 6: Synthetic Datasets Generation

dataset	total_samples	train_samples(n)	n_features(d)	depth	seed	thres	generate_data
SDI	40000	20000	20	4	365	0	through code
SDII	60000	30000	100	4	365	0	through code
SDIII	100000	50000	500	4	365	0	through code

This dataset is synthetically generated with specified dimensions from a labelling function f^* given by an Oblique Decision Tree (ODT) with depth and a defined number of data points as given in Table 6. The datapoints \mathbf{x} are drawn uniformly from the surface of a d -dimensional sphere of radius 1, centred at the origin. We used COB-ODTs as mentioned in Section 3 with biases kept at zero. The leaf node labels are chosen so that sibling labels get opposite signs. The final output includes the pruned data, labels, and information about the tree’s structure. Three synthetic datasets (SD) are used, named SDI, SDII, and SDIII.

Table 6 presents the parameters used for constructing the datasets.

A.4.2 TABULAR DATASETS:

We used a total of 20 tabular binary classification datasets for the comparative study of our models. Most datasets are available in the UCI repository <https://archive.ics.uci.edu/datasets>. Some are taken from the OpenML benchmark, as given in the paper (Grinsztajn et al., 2022). https://www.openml.org/search?type=benchmark&study_type=task&sort=tasks_included&id=298. The dataset download URL is in Table 7. After downloading the datasets, they are preprocessed by dropping rows with missing values, converting categorical features using Label Encoding, Standardizing numerical features, and Encoding the target variables to 0 and 1.

Figure 5 illustrates an example scenario when a 3-hidden layer DLGN is trained on data given in Figure 1(c). The initial hyperplanes(h-planes) given by V^1 and V^2 and V^3 as shown in Figure 5(a-c) are essentially random. However, after training, the hyperplanes in the later layers show a remarkable tendency to move towards the hyperplanes corresponding to the decision tree – particularly that of nodes close to the root (See Figures 5(d-f)).

A.5 EXPERIMENTAL SETUP DETAILS

A.5.1 TRAIN_VALIDATION_TEST SPLIT:

For the synthetic datasets SDI, SDII, and SDIII, the dataset is split into 50% train, 25% test, and 25% validation set. Models are trained on the training data and validated on the validation set, and

Table 7: Tabular datasets

data	download_link
Adult	https://archive.ics.uci.edu/static/public/2/adult.zip
Bank	https://archive.ics.uci.edu/static/public/222/bank+marketing.zip
Card	https://archive.ics.uci.edu/static/public/350/default+of+credit+card+clients.zip
Telesc	https://archive.ics.uci.edu/static/public/159/magic+gamma+telescope.zip
Rice	https://archive.ics.uci.edu/static/public/545/rice+cammeo+and+osmancik.zip
Stat	http://archive.ics.uci.edu/static/public/144/statlog+german+credit+data.zip
Spam	http://archive.ics.uci.edu/static/public/94/spambase.zip
Gyro	https://archive.ics.uci.edu/static/public/755/accelerometer+gyro+mobile+phone+dataset.zip
Swar	https://archive.ics.uci.edu/static/public/524/swarm+behaviour.zip
Credit	https://api.openml.org/data/v1/download/22103185/credit.arff
Elec	https://api.openml.org/data/v1/download/22103245/electricity.arff
Cover	https://api.openml.org/data/v1/download/22103246/covertime.arff
Pol	https://api.openml.org/data/v1/download/22103247/pol.arff
House	https://api.openml.org/data/v1/download/22103248/house_16H.arff
Mini	https://api.openml.org/data/v1/download/22103253/MiniBooNE.arff
Diab	https://api.openml.org/data/v1/download/22111908/Diabetes130US.arff
Jannis	https://api.openml.org/data/v1/download/22111907/jannis.arff
Bior	https://api.openml.org/data/v1/download/22111905/Bioresponse.arff
Calif	https://api.openml.org/data/v1/download/22111914/california.arff
Heloc	https://api.openml.org/data/v1/download/22111912/heloc.arff

then the test score is reported against the test data with the best hyperparameters. Similarly, for the tabular datasets, the dataset is split into 60% train, 20% test, and 20% validation set.

A.5.2 NUMBER OF FOLDS:

Based on the algorithms used, the number of folds used is also varied. For standard ML algorithms like CART, Random Forest, SVM, and SVM Linear, we use 3-fold cross-validation; for most other algorithms, including DLGNs, we use one-fold size.

A.5.3 HARDWARE:

All the experiments are performed on Kaggle, and all the neural network-based experiments use GPU, whereas traditional ML algorithms are performed on CPU. Kaggle provided GPUs, such as GPU T4 x2 and GPU P100.

A.5.4 HYPERPARAMETERS TUNING:

Each algorithm used in this paper has a different set of hyperparameters, and hyperparameter tuning is one of the most important aspects for getting the best accuracy. Here, for each algorithm, we extensively searched from a collection of hyperparameters and validated their result on the validation set to obtain the best-performing hyperparameters. Test results are reported based on the best hyperparameters. The below tables give a list of all hyperparameters for each algorithm used.

Table 8: DLGN and DLGN-SF hyperparameters space

Parameters	Set of values searched on
num layers	3, 4, 5
num nodes(each layer)	10, 20, 50, 100, 200, 500, 1000
beta	3, 10, 20, 30
learning rate(lr)	0.1, 0.01, 0.02, 0.05, 0.001, 0.005, 0.0001
epochs	200, 300, 500
batches	1, 10, 100
optimizers	Adam, SGD
weight decay	0, 0.1

Table 8 contains the hyperparameter set used in training the DLGN and DLGN-SF models on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. Parameters num layers, num nodes(each layer), and beta are the most important hyperparameters.

Table 9: DLGN-VT hyperparameters space

Parameters	Set of values searched on
num layers	3, 4
num nodes(each layer)	10, 20
beta	3, 10, 20, 30
learning rate(lr)	1., 0.1, 0.01, 0.02, 0.05, 0.001, 0.005, 0.0001
epochs	200, 300, 500, 1000, 5000, 10000, 15000
batches	1, 10, 100
optimizers	Adam, SGD
weight decay	0, 0.1
C	0.1, 0.03, 1.0
max_iter	100, 500, 1000
penalty	l1, l2
solver	liblinear

Table 9 contains the hyperparameter set used in training the DLGN-VT model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. Along with the DLGN parameters, it has C, max_iter, penalty and solver as additional parameters.

Table 10 contains the hyperparameter set used in training the DLGN-DT models on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. It has eps, min_samples and max_depth as important parameters for finding the cluster centre and depth of the tree constructed.

Table 11 contains the hyperparameter set used in training the ReLU models on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy.

Table 12 contains the hyperparameter set used in training the Linear SVM models on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. Here, kernel is set to Linear for Linear SVM.

Table 10: DLGN-DT hyperparameters space

Parameters	Set of values searched on
num layers	3, 4, 5
num nodes(each layer)	10, 20, 50, 100, 200, 500, 1000
beta	3, 10, 20, 30
learning rate(lr)	0.1, 0.01, 0.02, 0.05, 0.001, 0.005, 0.0001
epochs	200, 300, 500
batches	1, 10, 100
optimizers	Adam, SGD
weight decay	0, 0.1
eps	0.1, 0.2, 0.3
min_samples	1, 2, 3, 5, 7, 10, 15, 22, 40
max_depth	1 to 10

Table 11: ReLU hyperparameters space

Parameters	Set of values searched on
num layers	3, 4, 5
num nodes(each layer)	10, 20, 50, 100, 200, 500, 1000
learning rate(lr)	0.1, 0.01, 0.02, 0.05, 0.001, 0.005, 0.0001
epochs	200, 500, 1000
optimizers	Adam, SGD
weight decay	0, 0.1

Table 13 contains the hyperparameter set used in training the Non-linear SVM models on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. Here, the kernel is set to Non-linear Kernels like rbf or sigmoid.

Table 14 contains the hyperparameter set used in training the CART model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. max_depth is the most important hyperparameter to train.

Table 15 contains the hyperparameter set used in training the random forest model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. n_estimators define the number of estimators in the random forest as one of the most vital hyperparameters.

Table 16 contains the hyperparameter set used in training the SDT model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. depth is the tree depth of SDT which is the most vital hyperparameter to tune.

Table 17 contains the hyperparameter set used in training the TAO model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. n_iters, max_leaf_nodes and min_node_samples are important hyperparameters to train.

Table 18 contains the hyperparameter set used in training the Zan-DT models on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. depth, reg and mlp_layer are vital parameters.

Table 19 contains the hyperparameter set used in training the Disnn model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which is then used to get the test accuracy. A value of 10-15 for n_polytopes_list and m_list works best.

Table 20 contains the hyperparameter set used in training the GLN model on all the datasets. The trained model is tested on a validation set, and the best hyperparameter combination is found, which

Table 12: SVM Linear hyperparameters space

Parameters	Set of values searched on
C	0.1, 0.5, 1, 2, 5
kernel	Linear

Table 13: SVM hyperparameters space

Parameters	Set of values searched on
C	0.1, 0.5, 1, 2, 5
kernel	rbf, sigmoid
gamma	scale, auto, 0.001, 0.01, 0.1, 1, 10
degree	2, 3, 4, 5

is then used to get the test accuracy. layer_sizes, num_nodes and context_map_size are important parameters. context_map_size = 4 gives the best result.

Table 14: CART hyperparameters space

Parameters	Set of values searched on
criterion	gini, entropy
splitter	best, random
max_depth	1 to 10
min_samples_split	1 to 10
min_samples_leaf	1, 2, 4, 5
max_features	sqrt, log2

Table 15: Random Forest hyperparameters space

Parameters	Set of values searched on
criterion	gini, entropy
n_estimators	10, 20, 50, 100
max_depth	1 to 10
min_samples_split	1 to 10
min_samples_leaf	1, 2, 4, 5
max_features	sqrt, log2

Table 16: SDT hyperparameters space

Parameters	Set of values searched on
depth	1 to 10
lamda	0.1, 0.01, 0.02, 0.05, 0.001
learning rate(lr)	0.1, 0.01, 0.02, 0.05, 0.001, 0.005, 0.0001
epochs	200, 300, 500
batches	32, 64, 128
weight decay	0, 0.1, 0.0005

Table 17: TAO hyperparameters space

Parameters	Set of values searched on
n_iters	10, 20, 30
max_leaf_nodes	5, 10, 15
randomize_tree	True, False
update_scoring	accuracy
min_node_samples_ao	1, 2, 3, 4, 5
min_leaf_samples_ao	1, 2, 3, 4, 5
reg_param	0.1, 0.01, 0.02, 0.05, 0.001

Table 18: Zan-DT hyperparameters space

Parameters	Set of values searched on
depth	1 to 10
reg	0.1, 0.01, 1.48, 1.5, 2
mlp_layer	3, 4, 5
dropout	0.0, 0.01, 0.05, 0.07, 0.1
lr	0.1, 0.01, 0.02, 0.05, 0.001
epochs	200, 300, 500
batches	32, 64, 128, 512

Table 19: Disnn hyperparameters space

Parameters	Set of values searched on
n_polytopes_list	1 - 15,20,30,100
m_list	1 - 15,20,30,100

Table 20: GLN hyperparameters space

Parameters	Set of values searched on
layer_sizes	3,4,5
num_nodes	5,10,20,50
context_map_size	2 to 10
lr	0.1, 0.01, 0.02, 0.05, 0.001,0.00025