M-Boost: Profiling and Refining Deep Neural Networks with Topological Data Analysis*

Extended Abstract†

Gregory Naitzat
University of Chicago
Chicago, Illinois
gregn@galton.uchicago.edu

Namita Lokare
SAS Institute, Inc.
Cary, North Carolina
namita.lokare@sas.com

Jorge Silva
SAS Institute, Inc.
Cary, North Carolina
jorge.silva@sas.com

Ilknur Kaynar-Kabul
SAS Institute, Inc.
Cary, North Carolina
ilknur.kaynarkabul@sas.com

ABSTRACT

This extended abstract reports work in progress on a topology-based approach to the problem of profiling, diagnosing and refining black-box models, with particular emphasis on deep neural networks. The proposed method is named M-Boost and relies on the mapper algorithm from topology, recursively identifying groups of observations where the accuracy can be improved.

The advantages of M-Boost are in its conceptual simplicity and universality: (i) it is universal for any network architecture and data set type, whether image-based or otherwise; (ii) it has an inherent notion of resolution going from coarse to fine detail; (iii) it provides visualization which is straightforward to read and interpret by non-experts.

The extended abstract includes a detailed description and experimental results with real data on credit card payment defaults.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; **Neural networks**;

KEYWORDS

Deep learning, interpretability, topological data analysis

ACM Reference Format:
Gregory Naitzat, Namita Lokare, Jorge Silva, and Ilknur Kaynar-Kabul. . M-Boost: Profiling and Refining Deep Neural Networks with Topological Data Analysis: Extended Abstract. In *Proceedings of prepring (KDD)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

*Produces the permission block, and copyright information
†The full version of the author's guide is available as `acmart.pdf` document

KDD, 2018,

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1 INTRODUCTION

In recent years, much work has been devoted to understanding what a "black box" model is actually doing. The emerging field of interpretability strives to understand the reasoning of machine learning systems. As machine learning algorithms are adopted in a growing number of applications, the demand for tools to help interpret and gain confidence in their performance is expected to continue to grow. This is especially true with the recent advancements in deep neural networks (DNNs), given their complexity. The terms "interpretability", "reasoning", "confidence" are rather imprecise, blending a variety of concepts and techniques. Some techniques can be qualitative —we can reason about performance of a DNN by looking at visual depictions of the model inner representations, identify important regions in the input or subset of crucial parameter values; other techniques are quantitative —where we seek evidence by using some metric (*e.g.* accuracy).

The drawback of qualitative methods is that they are highly subjective and require human involvement. Nevertheless, qualitative analysis, seems to be the main currently available way to interpret any given neural network and to gain more confidence in its performance. Zeiler et. al.[20] have introduced a visualization technique that gives insight into the intermediate layers. The authors use this tool to analyze the visualizations and then, based on visual understanding, they modify parameters such as stride length and filter size to improve performance. Yosinski et. al.[19] introduce a toolbox to plot activations produced on each layer and features computed by individual neurons. These visualizations help understand which features are learned by individual neurons. The difficulty with those and other approaches is that such investigation techniques tend to be domain specific, require a bulky setup, and often rely on specialized or heavily annotated data sets. Moreover, the interpretation framework is often more complicated than the examined DNN itself, and therefore the construction of a "test bench" for analysis of a DNN might be too large of an effort.

The method proposed in this work leverages qualitative and quantitative approaches: we use topological data analysis to qualitative summarize neural network's inner representations and we use quantitative metrics to profile regions in the inner features space. Specifically, we visualize and analyze the output from each layer of the neural network. We profile the local regions that have

low accuracy scores and, importantly, we are able to further improve the DNN performance by using a local ensemble approach. Our method is conceptually simple and is applicable to models other than neural networks. We name the method M-Boost, where the M- refers to the mapper algorithm from topological data analysis, and Boost refers to ensembling models trained on local regions of the input space, for the purpose of enhancing performance.

We want to note a similar work [11] which is independent of our effort and of which the authors have became aware only recently. Unlike the cited work, we extend mapper visualization to intermediate layers and propose a boosting method.

This paper is organized as follows: Section 2 gives an overview of the tools that we used in our method, followed by our approach in Section 3. Experiments and results are discussed in Section 4, and conclusions are in Section 5.

2 BACKGROUND

In this section we describe the tools that we used in our analysis mainly neural networks (Section 2.1) the mapper algorithm (Section 2.2).

2.1 Neural networks

Neural networks have a long and rich history, which we do not detail here. One relevant work (among many) is LeCun et. al.[8] which applies neural networks handwritten digit classification. For decades, neural networks have been used in numerous applications, including but not limited to image classification, face detection and recognition, and speech recognition. Over the past few years, deep neural networks (DNN) have developed, and grown in terms of the number of layers and number of nodes per layer. For example, see AlexNet[7], GoogLeNet [17], VGG [15] and ResNet [6].

For the sake of establishing notation, we show a feed forward deep neural network below:

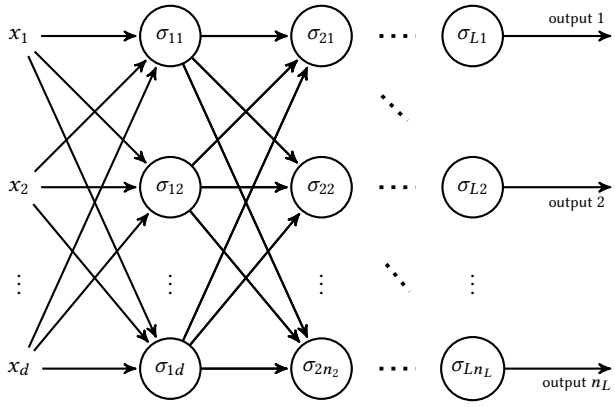

Figure 1: A feedforward neural network of L layers.

Each node, $\sigma_{ij}, i = 1, \ldots, L; j = 1, \ldots, k$, represents computation of a neural network, where a continuous nonlinear transformation $\sigma_{ij}(\cdot)$ is applied on an affine transformation of the input vector. To simplify our treatment we assume that $\sigma_{ij}(\cdot) \equiv \sigma(\cdot)$ and we treat each layer in a vectorized fashion, a single index $i \in \{1, \ldots, L\}$ is attached to a layer, the nonlinear transformation is applied coordinatewise, and the affine transformation in layer i is given by a matrix $W_{[i]}$, and a vector $b_{[i]}$ of appropriate dimensions to match the input and the output widths of the ith layer. Overall, a layer i operating on its input, $x_{[i]}$, is compactly written as

$$\sigma_{[i]}(x_{[i]}) := \sigma\left(W_{[i]}x_{[i]} + b_{[i]}\right).$$

The complete network is thus given by the continuous function $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^{n_L}$ of the form

$$\mathcal{N}(x) := \sigma_{[L]} \circ \cdots \circ \sigma_{[2]} \circ \sigma_{[1]}(x),$$

or explicitly

$$\mathcal{N}(x) = \sigma\left(W_{[L]}\sigma\left(W_{[L-1]}\sigma\left(\cdots\right) + b_{[L-1]}\right) + b_{[L]}\right).$$

Additionally, we denote the operation of the first i layers by $\mathcal{N}_{[i]}(x)$ and that of the last i layers by $\mathcal{N}_{[-i]}(x)$, i.e.,

$$\mathcal{N}_{[i]}(x) := \sigma_{[i]} \circ \cdots \circ \sigma_{[2]} \circ \sigma_{[1]}(x),$$
$$\mathcal{N}_{[-i]}(x) := \sigma_{[m]} \circ \cdots \circ \sigma_{[m-i+1]} \circ \sigma_{[m-i]}(x).$$

We set $\mathcal{N}_{[0]}(x) \equiv x$.

Note that by our numbering convention, the output of layer i is the input of layer $i + 1$, i.e.,

$$x_{[i+1]} = \mathcal{N}_{[i]}(x),$$

and $x_{[1]}$ denotes the initial input to the network; so (2.1) can be rewritten as

$$x_{[i+1]} = \mathcal{N}_{[i]}(x_{[1]}).$$

2.2 The Mapper Algorithm

The core of the proposed M-Boost technique is the mapper algorithm from topological data analysis (TDA). TDA is a recent development in the study of computational topology, the purpose of which is to develop topological and related concepts for data analysis and other "real-world" problems such as manifold sampling [10], target enumeration [1], topological signal processing [14], and many others. For exposition of the main aspects of topological data analysis we refer the reader to [5] [3] and [21].

Inspired by topological concept of Reeb graph (to learn more about Reeb graphs we refer to [2]), the mapper algorithm was introduced in [16] for point cloud visualization. It involves mapping data through a "*lens*" or "*filter*" function, which is an arbitrary function that maps high dimension point cloud to a low dimensional space. Perhaps the simplest example of a lens is a projection function $\ell(x) = \pi_i(x)$ mapping data point $x \in \mathbb{R}^d$ to its i-th coordinate $\pi_i(x) \mapsto x_i$. Different kinds of lens allow us to visualize different aspects of the data. In our work we select an unconventional lens —this lens is the output of the DNN. It turns the mapper algorithm into a visualization tool to investigate operation of a given DNN.

The mapper visualizes data as m-simplicial complex. A simplicial complex is a well known combinatorial object that can be geometrically realized by gluing simplices of dimensions 0 to m. A k-dimensional simplex in $\mathbb{R}^m$ is convex hull of $k + 1$ affinely independent points in $\mathbb{R}^m$. Namely, a zero dimensional simplex is a point, a one dimensional simplex is a line segments, two and higher dimension simplices are triangles, and their higher dimensional counterparts.

The simplices in the simplicial complex are allowed to intersect only along their facets and for any k dimensional simplex, the simplicial complex must also include all its faces as a separate simplices in their own right (see Figure 2 for illustration).

Construction of the simplicial complex by the mapper starts with a local clustering of the data, and aggregation of the clusters into a full shape based on the overlapping points between clusters. The local regions for clustering are determined by the lens - points with similar lens values are grouped, each of those groups is clustered separately. Thus, if two points are close in the input space but far apart under lens mapping, they will not be put in the same cluster. Therefore, a prerequisite for two data samples to be put in the same cluster is that their lens value is not too far apart.

With a slight abuse of notation we will represent a k simplex by listing the set of its vertices $(v_1, \ldots, v_{k+1})$ a zero simplex (a vertex) is then written as (v_i).

For clarity of exposition, from now on, we will assume that the lens function $\ell(\cdot)$ maps to n-unit square $[0,1]^n$. Setting $\ell(x) = \mathcal{N}(x)$, where $\mathcal{N}$ is a binary classifier network yields $\ell(x) \mapsto [0,1]$, if the lens is taken to be a ternary classifier, then we obtain $\ell(x)$ that maps to a subset of $[0,1]^2$ and so on.

Next, we give a step by step description of the mapper algorithm (for a complete description we refer to [16]). We start with the definition of *Nerve of a set*

Definition (Nerve of data point sets). *Let a finite collection of sets of data points $D_j \subset \mathbb{R}^d, j = 1, \ldots, K$ (i.e. each D_j is a collection of points). The nerve of sets $\{D_j\}_{j=1}^K$ is a simplicial complex constructed as the following:*

- *First, consider a finite collection of tuples of indexes such that a tuple $J_i = (j_1, \ldots, j_{|J_i|})$ is in the collection $N = \{J_i\}$ if and only if, the intersection of the corresponding sets $\{D_j\}$ is non empty, i.e:*

$$\bigcup_{j \in J_i} D_j \neq \emptyset.$$

- *The nerve of $\{D_j\}$ is the simplicial complex S formed by simplicies:*

$$\left\{(v_{i_1}, \ldots, v_{i_{|J|}}) \ : \ J \in N\right\} \cup \left\{(v_j)\right\}_{j=1}^K,$$

where v_j is an arbitrary vertex associated with each point set D_j.

Mapper algorithm, $\mathcal{M}$.

Input: Data point cloud $D = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d$; lens $\ell(\cdot) : \mathbb{R}^d \to [0,1]^n$; a clustering algorithm $C(\cdot) : \text{input} \to \{0, 1, \ldots, C\}$ (where c is a number of clusters, with $C = 0$ meaning no clusters were identified).

Result: Simplicial complex S.

(1) Slice the range of the lens $l(\cdot)$ into an overlapping regions

$$[0,1]^n = R_1 \cup \cdots \cup R_K.$$

For example, let $\ell(\cdot) \to [0,1]$. Then for $j = 1, \ldots, K$, take

$$R_j := \left[(j-1)(1/n) - \varepsilon, j(1/n) + \varepsilon\right] \cap [0,1],$$

where ε is small and determines how much the intervals overlap.

(2) For each region, R_j, find data points in D whose lens values fall within the region, i.e.

$$D_j := \{x \in D \ : \ \ell(x) \in R_j\}.$$

Number of regions R_j controls visualization resolution.

(3) Apply clustering algorithm C on each set of data points D_j and split data points according to which cluster they belong to. That it, setting $C_j \geq 0$ to be the number of the resulting clusters in D_j, obtain:

$$D_{jc} := \left\{x \in D_j \ : \ x \text{ in cluster c}\right\}, \ c = 1, \ldots, C_j.$$

(4) Simplicial complex S is a nerve of the sets

$$\{D_{11}, \ldots, D_{1C_1}, D_{12}, \ldots, D_{2C_2} \ldots, D_{K1}, \ldots, D_{KC_K}\}.$$

Return: Simplicial complex S.

The mapper algorithm can be thought of as a transformation of point cloud to a simplicial complex S:

$$\mathcal{M}(D) \mapsto S.$$

When lens is a scalar function the resulting simplicial complex is undirected graph. If we think of point cloud data as sampled from a high dimensional manifold, this graph can be interpreted as a discrete version of Reeb graph. Which retains certain topological features of the underlying manifold, for details we refer the reader to [4] and references there in.

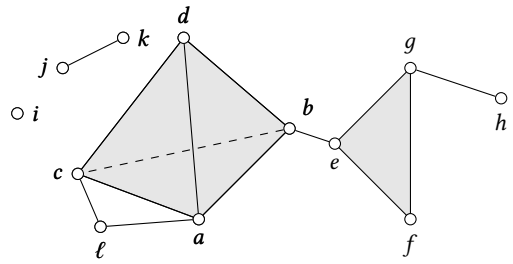

Figure 2: A 3 dimensional simplicial complex, (a, b, c, d) is 3-simplex, while (e, f, g) is 2-simplex as well as all the 2 dimensional facets of the 3-simplex, 1-simplices are formed by vertices $(e, b), (g, h)$, (c, ℓ), (ℓ, a) and (j, k) as well as all edges of the higher simplices, finally the zero simplices are the 12 vertices.

3 M-BOOST

We are now at the position to introduce M-Boost method. A single iteration of the method is conceptually described in Figure 3. The steps of the method are quite straightforward, however we hope that the formal exposition does not make them appear complicated and obscured. For clarity of exposition, we will consider a binary classification neural network. Using our notation from section 2

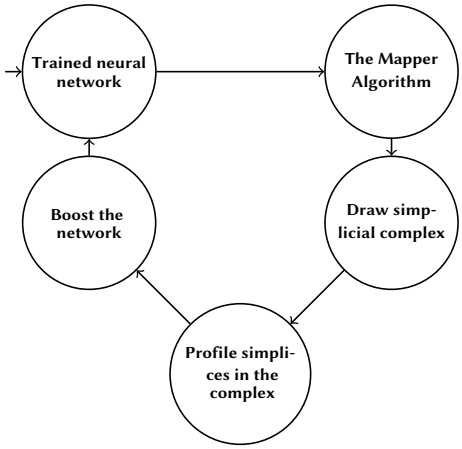

Figure 3: Conceptual summary of M-Boost technique, the drawing describes a work flow of a single iteration of the method.

the M-boost technique is described below.

M-Boost method, $\mathcal{B}$

Input: Validation set $D = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d$; trained binary classifier neural network $\mathcal{N}(\cdot) : \mathbb{R}^d \to [0,1]$; a decision tree $\mathcal{T}(\cdot)$: input $\to \{0,1\}$; the mapper algorithm $\mathcal{M}$.

Result: Improved classifier $\mathcal{B}$.

(0) Initialize $\mathcal{B} \leftarrow \mathcal{N}$.
(1) For a desired layer $k = 1, \ldots, L$ map
$$\{x_i\}_{i=1}^N \mapsto \{\mathcal{N}_{[k]}(x_i)\}_{i=1}^N.$$
(2) Apply $\mathcal{M}$ on point cloud $\{\mathcal{N}_{[k]}(x_i)\}_{i=1}^N$ to obtain simplicial complex. Since $\mathcal{N}$ has scaler output, the resulting complex is a graph $G = (V, E)$.
(3) For each node $v \in V$, calculate the cross entropy error $v_H := -\sum_{x_i \in v} \mathbb{1}_A(x_i) \log \mathcal{N}(x_i)$ where $\mathbb{1}_A(x_i) = 1$ when x_i belongs to class A and zero otherwise. The value of $\mathcal{N}(x_i)$ is interpreted as the predicted probability to belong to class A.
(4) For a desired threshold $t > 0$, denote $V_- := \{v \in V | v_H > t\}$ and $V_+ := V \setminus V_-$.
(5) Use the structure of complex S (graph G) to profile nodes in V_- and V_+ and select features that best distinct between those nodes. This is a crucial step which makes M-Boost distinct from other ensemble methods. We will explain this step separately at the end of this subsection.
(6) Train a decision tree $\mathcal{T}(\cdot)$ on the identified features from the previous step to predict $x_i \in V_+$ or $x_i \in V_-$ (when $\mathcal{T}(x_i) = 1$ or $\mathcal{T}(x_i) = 0$).
(7) Regenerate training and validation sets, split the training set into two separate data sets according to the prediction of the decision tree $\mathcal{T}(\cdot)$ obtained above. Re-train two neural networks on the separated training data sets.
(8) Stronger classifier $\mathcal{B}$ is obtained by replacing $\mathcal{N}$ with the two neural networks and the decision tree (that decides which of the two networks to activate).
(9) Repeat from step (1), on each newly appended network in $\mathcal{B}$ until no node in the generated graph appears below a threshold t or there is no improvement in the performance of the resulting model.

Return: Classifier $\mathcal{B}$.

Figure 4 illustrates the original network, and the output of the method after one cycle and Figure 5 illustrates the output at the end of second iteration of the method.

Now we expand on step (5), graph G (or in general the simplex S) represents topology of the point cloud data after it has been transformed by the neural network. If the network were perfectly trained, we would expect to see only two distinct clusters, one per class. In general, the network is not perfectly trained in this sense, and therefore we see variations in the shape of the clusters. This way we can understand what information the neural network captures and what information is not used – this is key in making sense of the operation of the neural network at a local level. We investigate different nodes on the graph and look for nodes with poor performance. The location of the poorly performing regions (V_-) and the good performing regions (V_+) on the graph G is taken into account in our profiling in step step (5) of the method. A simple way to do this profiling is to color the graph according to different features values. Coloring that come up aligned with nodes V_- and V_+ can provide for the desired features for boosting. This feature selection approach is similar to the one describe in [9], where the authors profile the output of the mapper to find patterns among groups of cancer patients. Combinatorial structure of the simplicial complex allows for potential future automation of the profiling step.

Before we conclude this subsection we want to stress one last point: while we have limited our treatment to network with scaler output and therefore the simplicial complex is of maximal dimension one, i.e. a graph, the method applies to a vector valued networks that can produce simplicial complexes of higher dimensions.

3.1 Methodology

In this sections we discuss some practical details related to the visualizations of neural networks and M-Boost.

The simplicial complex produced by the mapper algorithms summarizes the shape of point cloud, we can investigate the shape directly on the input data or we can slice the neural network and look on its inner representations . We found it most useful to investigate top layers in the neural network (which amount so setting $k \approx L$ in step (1) of the Mapper) those have lower dimensionality than the bottom layers and the representation in those layer is more consolidated. However it is interesting to examine the shape of other inner layers as well and trace how the representation changes as we move up the layers. We will return to this point at the end of this subsection.

There are two main parameters for the mapper algorithm:

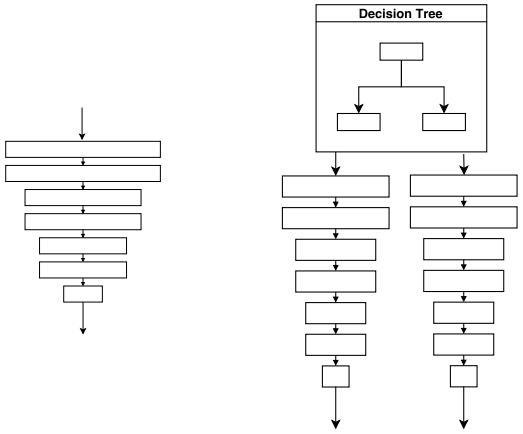

Figure 4: Left: the input to the M-boost method. Right: the output of the method after one cycle.

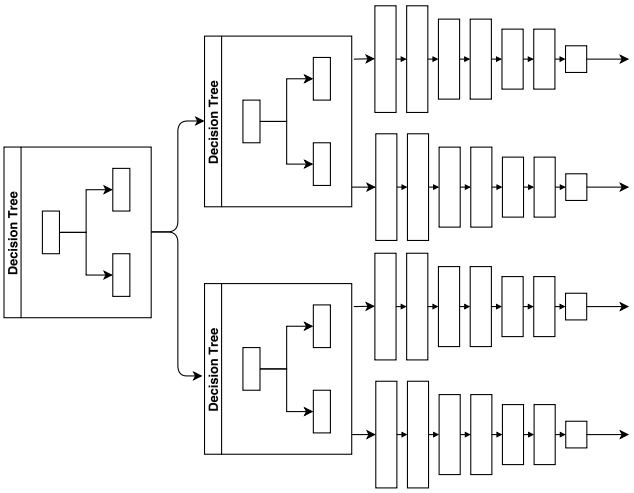

Figure 5: Output of the method after two cycles.

- Resolution - the number of slices in the domain of the lens (How changing resolution effects mapper output is illustrated in Figures 8 and 9, where the mapper is applied on the same input with the difference only due to change in the resolution. We will return to those figures shortly when we talk about the kind of visualization they illustrate.)
- Clustering radius – specific to the particular clustering algorithm selected (How different clustering radius effect operation of the mapper is show in Figure 10).

Once the radius and the resolution are appropriately selected, we can proceed to analyze the graph to select the feature to be used for boosting. To this end we perform the following:

- Apply coloring schemes to visualize how features of the input are mapped on the graph.
- Identify and attribute meaning to different regions on the graph.

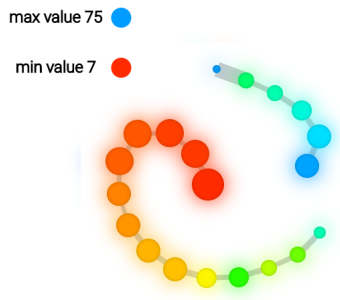

Figure 6: Output of the mapper representing the shape of the output for one layer before the final layer in the network, colored according to ground truth percentage of defaults.

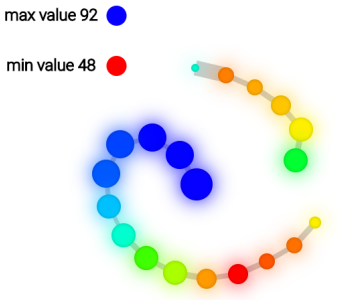

Figure 7: Same graph as in Figure 6, this time colored according to neural network prediction accuracy.

- Apply coloring schemes to trace how the performance of the neural network changes locally on the graph and how its performance is aligned with the distribution of the features.

To finalize this section we present another plot where we trace the change in the inner representation between different layers of the DNN. We select the node that we are interested in exploring, and trace its representations across four different layers. This is shown in Figures 8 and 9. The figures only differ in the selected resolution of the mapper algorithm, the red lines on the figure trace samples within the selected node starting from one layer before the final layer, then the fifth layer, the third layer and finally the first layer. The width of the line corresponds to the proportion of the samples shared by the nodes.

We can learn several things from this plot i) Qualitatively, moving up the layers we see consolidation of different nodes into one larger node ii) The shapes of the lower layer are more intricate they have more branching nodes, and remain fragmented for a larger range of clustering radius (this is also reflected in the tables 1 to 6, which summarize our selection of the clustering radius for different layers, observe that in the top layers the change in the number of cluster is gradual while in bottom layers the change is abrupt).

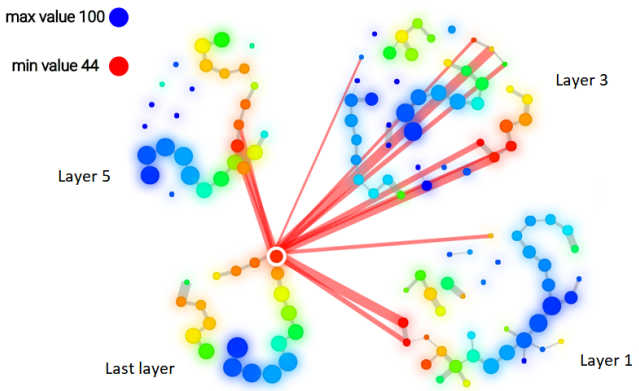

Figure 8: Samples tracing across different layers. The coloring reflects the accuracy of the neural network prediction. Width of the line reflect the number of samples shares between nodes.

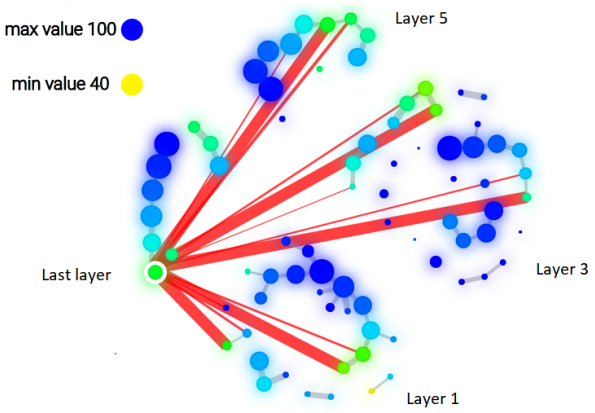

Figure 9: Samples tracing across different layers, same plot as in Figure 8 but with lower resolution, i.e. the range of the lens is split into less regions.

4 EXPERIMENTS

4.1 Dataset

We use the credit card default payment dataset from [18]. The data set consists of cardholder data from a bank in Taiwan. This data set has a total of approximately 25,000 observations, of which 5529 observations correspond to the card holders with defaults. The binary response variable correspond to defaulted payment (Yes = 1, No = 0). The authors of the cited study compare different modeling techniques, and report (= 0.17) error rate for a neural network model which preforms best among the compared data mining techniques: Neural networks, Decision trees, Naive Bayesian Classifier, Discriminant analysis, logistic regression, K-nearest neighbors classifier.

We extract 210 features from the existing 23 explanatory variables. The dataset is split randomly into training, validation and test set (0.4, 0.4, 0.2 ratio).

We build a feedforward neural network for prediction of the default payments and train it on the data set. The feedforward network consists of 7 layers ($L = 7$) of varying width. The last two layers (layer 7 and 6) are activated by softmax non-linearity while the other layers are activated with ReLU non-linearity. Writing N for the width of the first layer, i.e. $n_1 = N$, the width of the subsequent layers is adjusted as the following $n_2 = N, n_3 = n_4 = \frac{N}{4}, n_5 = n_6 = \lfloor \frac{N}{25} \rfloor$, and the final layer has only one unit. We report or experimentation results for the case where $N = 512$.

4.2 Mapper parameters

We apply the M-boost algorithm based on the output of one layer before the final output of the neural networks ($\{\mathcal{N}_{[-1]}(x_i)\}$) and pick the lens to be $\ell(\cdot) = \mathcal{N}(\cdot)$. We start our investigation with a very aggressive clustering (small clustering radius). This produces a highly fragmented graphs in the output of the mapper step, as we increase the radius, two distinct stable clusters appear. This is illustrated in Figure 10. The coloring of the figure corresponds to the performance of the neural network going from blueish to reddish, where red corresponds to lower accuracy.

We can glean a few observations from this short analysis: i) The larger cluster corresponds mostly to the non-default population. The smaller cluster corresponds to the default population. ii) Accuracy of the model on default cluster is worse than on non default cluster. The non-default cluster goes from high accuracy in one end of the cluster to lower accuracy towards the other end of the cluster. At our next step, we will profile the nodes with weak accuracy in order to find out which features can be used to boost the performance of the DNN.

4.3 Profiling

a We focus on the two stable clusters (see Figures 6, 7) and apply coloring scheme that is based on the ground truth - blue is where the default probability is low, - red is for clusters with high default probability, this is shown in Figure 6. We profile the clusters that have higher mis-classification rate by using LIME [13] and standard statistical methods (bar plots). But, we are particularly interested in finding features which might not be picked by traditional feature selection. From the LIME plots we saw that gender has a significant weight that decides whether the prediction is default or non-default. We also notice that gender does not come up as significant feature after using sklearn [12] to select the best features from this data set (based solely on explained variance, age and gender appear at place 53 and 65 in the features importance list, respectively).

This suggest that we might be able to use age as a boosting feature to enhance the model (how the age is locally aligned with the graph is shown in Figure 16). With additional experimentations with different color schemes we have picked gender as another candidate feature to be used for boosting (see Figure 11). In the next section we describe how we apply M-Boost.

4.4 M-Boosting

Here we demonstrate application of the M-Boost as described in section 3. We use the setup describe in section 4.2. Clustering radius is set to produces two distinct isolated clusters as on Figure 11. For boosting we train the decision tree $\mathcal{T}(\cdot)$ to predict default or non-default outcome based solemnly on the features picked by local

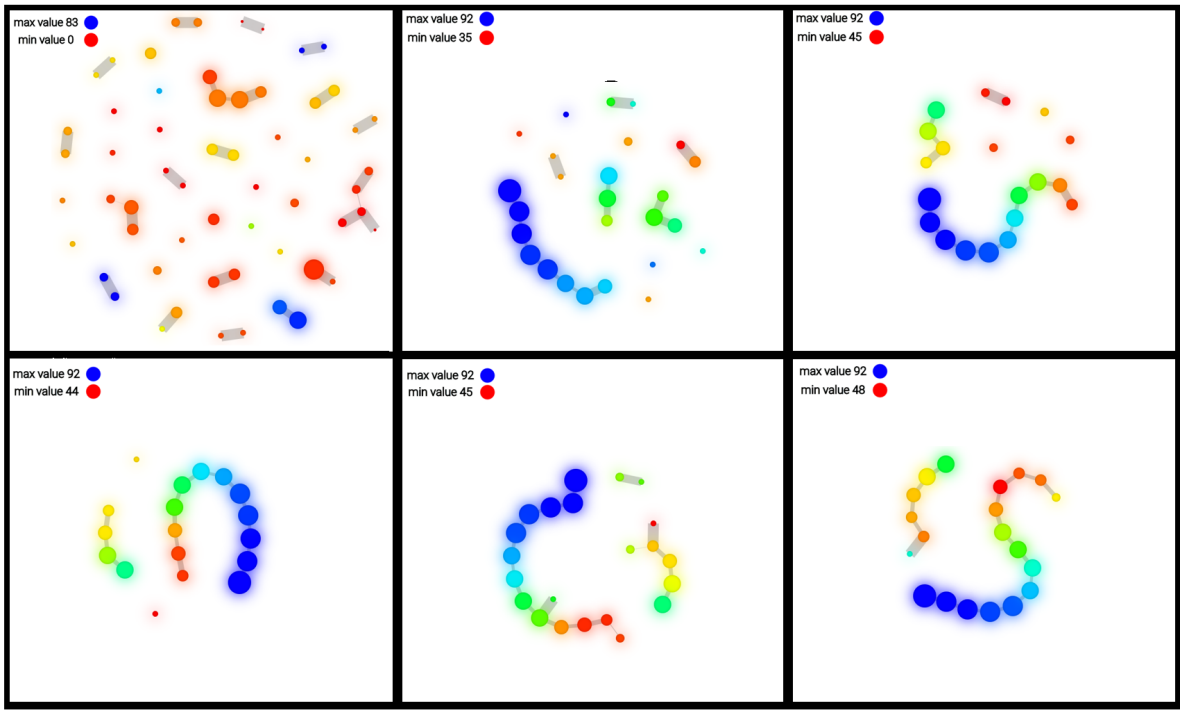

Figure 10: Inner representation for decreasing clustering radius. Top left an aggressive clustering (small clustering radius), bottom right a mild clustering (large clustering radius). The coloring of the figure corresponds to the accuracy of the neural network.

Table 1: Layer 1

radius	# of cluster
0.01	35
0.015	12
0.02	5
0.025	4
0.03	3
0.035	3
0.04	2

Table 2: Layer 2

radius	# of cluster
0.25	54
0.3	12
0.35	7
0.4	5
0.45	3
0.5	3
0.55	2

Table 3: Layer 3

radius	# of cluster
0.35	31
0.4	16
0.45	4
0.5	9
0.55	7
0.6	5
0.65	3

Table 4: Layer 4

radius	# of cluster
0.4	25
0.5	29
0.6	27
0.75	21
0.8	8
0.85	6
0.9	4

Table 5: Layer 5

radius	# of cluster
0.85	70>
0.9	61
0.95	44
1.0	28
1.1	13
1.2	8
1.3	12

Table 6: Layer 6

radius	# of cluster
1.5	15
1.6	14
1.7	14
1.8	13
1.9	12
2.0	16
2.1	6

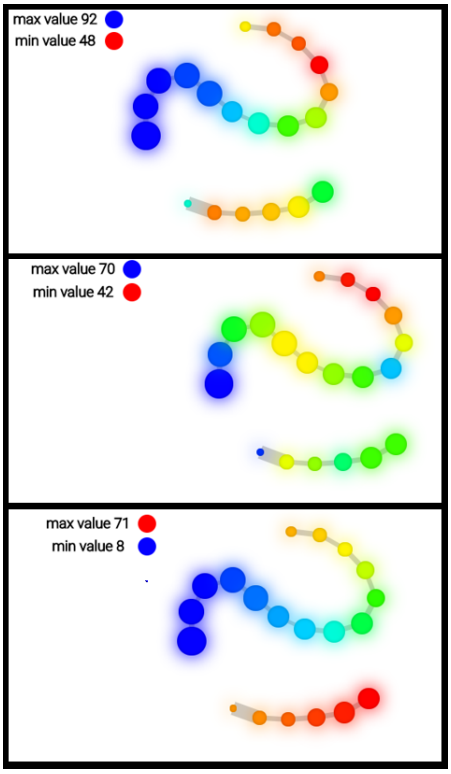

Figure 11: Profiling nodes with lower accuracy from top to bottom: Accuracy, percentage of females, predicted probability for default.

profiling of weakly performing nodes that we discussed at in the previous section.

4.4.1 Boosting by age. The run of the M-boost method produces the results shown in Figure 13.

4.4.2 Boosting by gender. The run of the M-boost method produces the output shown in Figure 15.

The plots referenced above show an improvement in the frequency of well-classified observations for the boosted model, when boosting by either age or gender (locally).

5 CONCLUSION

We have described work in progress on profiling, diagnosing and improving deep neural networks based on topological data analysis. Using the mapper algorithm, we are able to generate meaningful visualizations of the output of each layer. We automatically identify groups of observations that have the highest mis-classification, and refine the model by constructing an ensemble of locally valid networks in a recursive and automated manner. The method is simpler and more universal than other interpretability frameworks. We have validated the proposed method, named M-Boost, on real data and obtained promising results. For further work, we intend to conduct broader experiments on a wider variety of data sets. Additionally, we hope to glean deeper insights from tracing nodes of the topological graphs (obtained via the mapper) across each successive network layer. Finally, we plan to adapt the technique for use with other classes of black-box models besides DNNs, namely gradient boosting and random forest models.

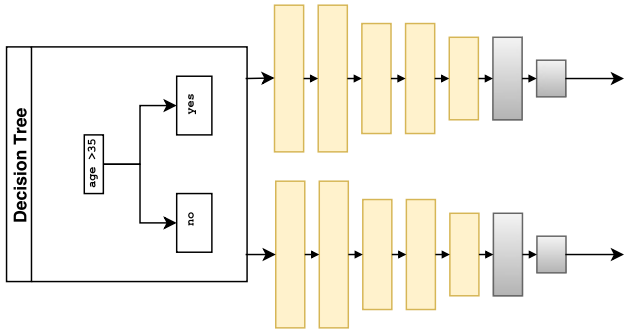

Figure 12: M-boost devised solution, decision tree splits the input according to age

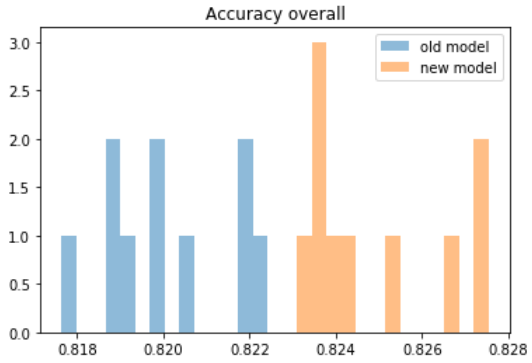

Figure 13: Histogram for comparison of the accuracy of the initial model vs boosted model (boost by age)

REFERENCES

[1] Yuliy Baryshnikov and Robert Ghrist. 2009. Target enumeration via Euler characteristic integrals. *SIAM J. Appl. Math.* 70, 3 (2009), 825–844.
[2] Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. 2008. Reeb graphs for shape analysis and applications. *Theoretical Computer Science* 392, 1-3 (2008), 5–22.
[3] Gunnar Carlsson. 2009. Topology and data. *Bull. Amer. Math. Soc.* 46, 2 (2009), 255–308.
[4] Mathieu Carriere and Steve Oudot. 2017. Structure and stability of the one-dimensional mapper. *Foundations of Computational Mathematics* (2017), 1–64.
[5] R. Ghrist. 2014. *Elementary Applied Topology.* Createspace Independent Pub. https://books.google.com/books?id=Z5ATogEACAAJ
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.
[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems.* 1097–1105.
[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* 1, 4 (Dec. 1989), 541–551. https://doi.org/10.1162/neco.1989.1.4.541
[9] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. 2011. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational

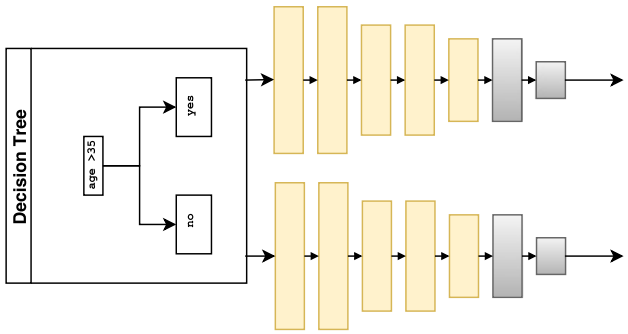

Figure 14: M-boost devised solution, decision tree splits the input according to gender

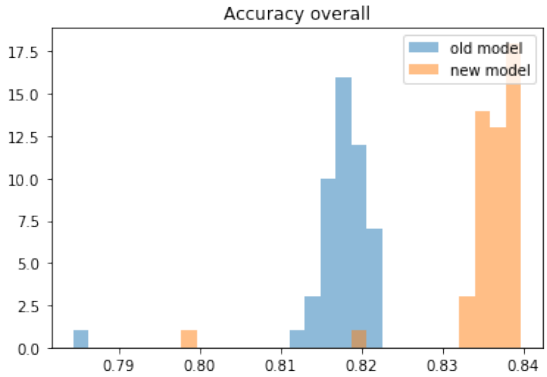

Figure 15: Histogram for comparison of the accuracy of the initial model vs boosted model (boost by gender)

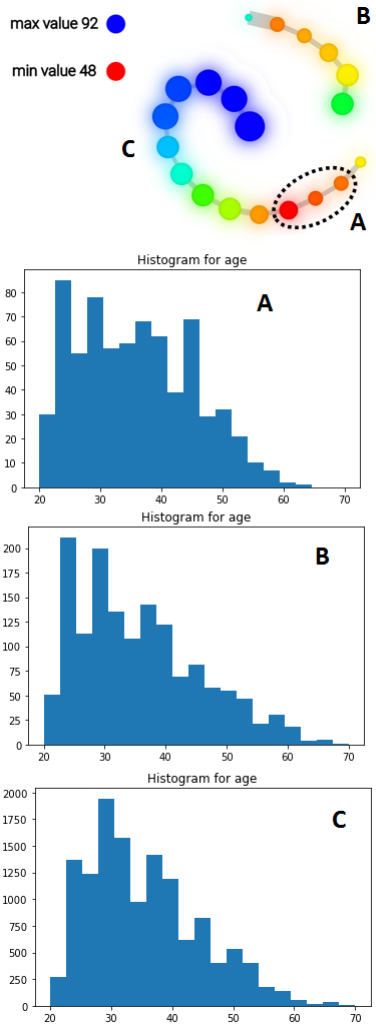

Figure 16: Age histogram profiling of nodes.

profile and excellent survival. *Proceedings of the National Academy of Sciences* 108, 17 (2011), 7265–7270.

[10] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. 2008. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry* 39, 1-3 (2008), 419–441.

[11] Paul T Pearson. 2013. Visualizing clusters in artificial neural networks using morse theory. *Advances in Artificial Neural Systems* 2013 (2013), 6.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[14] Michael Robinson. 2014. *Topological signal processing.* Springer.

[15] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[16] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. 2007. Topological methods for the analysis of high dimensional data sets and 3d object recognition.. In *SPBG*. 91–100.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[18] I-Cheng Yeh and Che-hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36, 2 (2009), 2473–2480.

[19] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[20] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[21] Afra J Zomorodian. 2005. *Topology for computing*. Vol. 16. Cambridge university press.