Noname manuscript No.
(will be inserted by the editor)

Auditing and Debugging Deep Learning Models via Flip Points: Individual-level and Group-level Analysis

Roozbeh Yousefzadeh · Dianne P. O'Leary

Received: date / Accepted: date

Abstract Deep learning models have been criticized for their lack of easy interpretation, which undermines confidence in their use for important applications. Nevertheless, they are consistently utilized in many applications, consequential to humans' lives, usually because of their better performance. Therefore, there is a great need for computational methods that can explain, audit, and debug such models. Here, we use *flip points* to accomplish these goals for deep learning classifiers used in social applications. A trained deep learning classifier is a mathematical function that maps inputs to classes. By way of training, the function partitions its domain and assigns a class to each of the partitions. Partitions are defined by the decision boundaries which are expected to be geometrically complex. This complexity is usually what makes deep learning models powerful classifiers. Flip points are points on those boundaries and therefore, the key to understanding and changing the functional behavior of models. We use advanced numerical optimization techniques and state-of-the-art methods in numerical linear algebra, such as rank determination and reduced-order models to compute and analyze them. The resulting insight into the decision boundaries of a deep model can clearly explain the model's output on the individual-level, via an explanation report that is understandable by non-experts. We also develop a procedure to understand and audit model behavior towards groups of people. We show that examining decision boundaries of models in certain subspaces can reveal hidden biases that are not easily detectable. Flip points can also be used as synthetic data to alter the decision boundaries of a model and improve their functional be-

Roozbeh Yousefzadeh
Yale University School of Medicine and VA Connecticut Healthcare System, New Haven, CT, USA 06510
E-mail: roozbeh.yousefzadeh@yale.edu

Dianne P. O'Leary
Computer Science Department and Institute for Advanced Computer Studies, University of Maryland, College Park, MD, 20742 E-mail: oleary@umd.edu

haviors. We demonstrate our methods by investigating several models trained on standard datasets used in social applications of machine learning. We also identify the features that are most responsible for particular classifications and misclassifications. Finally, we discuss the implications of our auditing procedure in the public policy domain.

Keywords Learning and adaptive systems (AMS: 68T05) · Reasoning under uncertainty (AMS: 68T37) · Deep learning · Interpretable machine learning · Auditing deep learning models

Declarations:

Funding

Not applicable.

Conflicts of interest/Competing interests

Not applicable.

Availability of data and material

All datasets used in the paper are public:
FICO Explainable Machine Learning Challenge dataset.[1]
Default of Credit Card Clients dataset [47].[2]
Adult Income dataset [5].[3]

Code availability

The code along with a readme file and an example procedure are available at `https://github.com/roozbeh-yz/auditing`.

Acknowledgments

The authors thank Brendan O'Leary and anonymous reviewers for helpful comments.

[1] `https://community.fico.com/s/explainable-machine-learning-challenge`
[2] `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`
[3] `https://archive.ics.uci.edu/ml/datasets/adult`

1 Introduction

Our focus in this paper is auditing and debugging deep learning models in social applications of machine learning. In these applications, deep learning models are usually trained for a specific task and then used, for example to make decisions or to make predictions. Despite their unprecedented success in performing machine learning tasks accurately and fast, these trained models are often described as black-boxes because they are so complex that their output is not easily explainable in terms of their inputs. As a result, in many cases, no explanation of decisions based on these models can be provided to those affected by them [51].

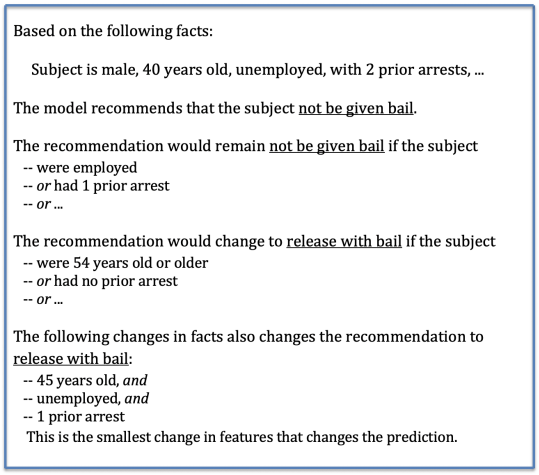

Fig. 1: Example of the kind of information that can be obtained by calculating flip points. We answer questions such as, "For a particular input to a deep learning model, what is the smallest change in a single continuous feature that changes the output of the model? What is the smallest change in a particular set of features that changes the output?"

This inexplainability becomes problematic when deep learning models are utilized in tasks consequential to human lives, such as in criminal justice, medicine, and business. Independent studies have revealed that many of these black-box models have unacceptable behavior, for example towards features such as race, age, etc. of individuals [37]. Because of this, there have been calls for avoiding deep learning models in high-stakes decision making [33]. Additionally, laws and regulations have been proposed to require decisions made by the machine learning models to be accompanied with clear explanations for the individuals affected by the decisions [43]. Several methods have been developed to explain the outputs of models simpler than deep learning models to non-expert users such as administrators or clinicians [18, 27, 34, 51]. In

contrast, existing interpretation methods for deep learning models either lack the ability to directly communicate with non-expert users or have limitations in their scope, computational ability, or accuracy, as we will explain in the next section.

In the meantime, deep learning is ever more widely used on important applications in order to achieve high accuracy, scalability, etc. Sometimes, deep learning models are utilized even when they do not have a clear advantage over simple models, perhaps to avoid transparency or to preserve the models as proprietary [35]. While it is not easy to draw the line as to where their use is advantageous, it is important to have the computational tools to thoroughly audit the models, provide the required explanations for their outputs, and/or to expose their flaws and biases. It would also be useful to have the tools to change their undesirable behavior.

Here, we take the view that a deep learning classifier is a mathematical function that maps inputs to classes. By training a model, we create a function that partitions the domain and assigns a class to each of the partitions. Partitions are defined by the decision boundaries which are expected to be geometrically complex. This complexity is what makes deep learning models powerful classifiers. To overcome these complexities, we use advanced numerical optimization techniques and state-of-the-art methods in numerical linear algebra.

1.1 Our contributions

The unique contribution of this work is to develop a comprehensive set of mathematical procedures, grounded in state-of-the-art methods in numerical linear algebra and optimization, in order to audit a trained model as a function. This sets a new standard on the scope and extent of auditing a trained model, and also on the proper mathematical formulations and procedures that should be used. We advocate for involvement of applied mathematicians in the public policy domain in setting the standards for auditing black-box models. We consider our paper as a step towards that goal.

In practice, there is a great need for auditing and debugging black-box models, most importantly, deep learning models. At the same time, the existing auditing methods in the literature are usually not comprehensive enough in their scope of analyzing the behavior of these models. Usually each paper focuses on one aspect of auditing while making their unique set of assumptions and simplifications.

The first component of our procedure is to formulate and solve optimization problems about trained models, based on the questions that an auditor has about the model. This has been considered to some extent in the literature, sometimes with shortcomings. Some of the shortcomings stem from the method of identifying points on the decision boundaries. Some studies use an ill-defined formulation to find the closest point on the decision boundaries. Some studies formulate the problem correctly, but resort to approximation methods to solve

it, which are not accurate enough as we have shown in our previous work [48]. Some other methods rely on simplifying assumptions about the decision boundaries. Some try to replicate the model's behavior with a simple model and then audit that simple model instead. We will review these methods in Section 2.

But, our contribution is not just formulating and solving optimization problems about trained models. We suggest exploring various sub-spaces in the domain of the function in order to provide the necessary understanding of the decision boundaries to an auditor, as formalized in our Algorithm 1. The type of feedback that our methods provide on the individual-level is illustrated in Figure 1 and discussed in Section 4.1. This will enable an auditor to obtain an accurate and thorough understanding of the model's behavior towards an individual, i.e., *individual-level auditing*. Our auditing does not stop there.[4]

Our other contribution is to use proper and well-understood methods in numerical linear algebra, such as rank determination and reduced-order methods, to audit the behavior of models towards groups of individuals, for example, people with certain race or certain education, i.e., *group-level auditing*. This is a concept well-known in social studies, but less considered as a formal procedure for auditing black-box models. We formalize this procedure in our Algorithm 2. We also use flip points to debug a model by altering the model's decision boundaries and changing its behavior towards groups of individuals.

1.2 Our plan

In Section 2, we review the literature and explain the advantages of our method compared to other popular methods such as LIME [31]. In Section 4, we present our computational approach to perform the above tasks, based on investigating and altering the decision boundaries of deep learning models by computing *flip points*, certain interesting points on those boundaries, defined in Section 3, where we also introduce the concept of constrained flip points. In Section 5, we present our numerical results on three different datasets with societal context. Section 6 compares our methods with other applicable methods in the literature. In Section 7, we present our conclusions and directions for future work, and finally, in Section 8, we discuss the policy implications of our work.

2 Literature review

There have been several approaches proposed for interpreting deep learning models and other black-box models. Here we mention a few papers representative of the field.

Spangher et al. [40] (independently) defined *a flip set* as the set of changes in the input that can flip the prediction of a classifier. However, their method is

[4] Although our examples involve neural networks, our methods are applicable to general, even black-box, classifiers.

only applicable to linear classifiers such as linear regression models and logistic regression. Similarly, Wachter et al. [44] define *counterfactuals* as the possible changes in the input that can produce a different output label and use them to explain the decision of a model. However, their mathematical approach is ill-defined since they do not consider the decision boundaries and as a result, the solution to their formulation cannot yield the closest point on the decision boundary. We present their formulation and explain this further in Section 3.3. Moreover, their proposed algorithm uses enumeration, applicable only to a small number of features. Russell [38] later suggested integer programming to solve such optimization problems, but the models used as examples are linear with small dimensionality, and the closest counterfactual in their formulation is ill-defined.

The approaches above use flip sets and counterfactuals to explain the least changes in individual inputs but do not go further to interpret the overall behavior of the model or to debug it. Moreover, their computational approaches are not applicable for deep learning. Although the idea of using counterfactuals is useful as a concept, it is important to relate them to the decision boundaries of the models, because as we noted earlier, a trained model is a mathematical function defined by its decision boundaries.

There are a few other studies on counterfactuals. For example, Mothilal et al. [26] use a similar ill-defined formulation as Wachter et al. [44] to compute counterfactuals for a given input, then train a second model on the counterfactuals to approximate the location of decision boundaries. Training a second model to approximate the decision boundaries adds another layer of uncertainty to their approach. Then they sample 1,000 random points in the neighborhood of the obtained point. They report that sometimes they cannot find any point on the decision boundaries. In our approach, we solve a well-defined optimization problem and find exact points on the boundaries.

Some studies have taken a model-agnostic approach to interpreting black-box models. For example, the approach taken by Ribeiro et al. [31], known as LIME, randomly perturbs an input until it obtains points on two sides of a decision boundary and then performs linear regression to estimate the location of the boundary in that vicinity. The simplifying assumption to approximate the decision boundary with hyperplanes can be misleading for deep learning models, as shown by Fawzi et al. [9] and Yousefzadeh and O'Leary [49]. Hence, the output of the LIME model and its corresponding explanation may actually contradict the output of the original model, as empirically shown by White and Garcez [46]. Another issue in LIME's approach is the reliance on random perturbations of inputs, which has computational limitations. Lakkaraju et al. [21] have also shown via surveys that such explanations may not be effective in communicating with non-expert users. Our method has an accuracy advantage over LIME, because we find a point exactly on the decision boundary instead of estimating its location via a surrogate linear regression model. Additionally, our explanation report can directly communicate with non-expert users such as credit applicants or clinicians.

There are approaches that create rule-lists based on the classifications of a deep learning model, and then use the obtained rules to explain the outputs [20, 21, 32]. These approaches have serious limitations in terms of scalability and accuracy, mostly because a deep learning model is usually too complex to be emulated via a simple set of if-then rules. For example, the outputs of the if-then rules obtained by Lakkaraju et al. [21] are different than the outputs of their neural network for more than 10% of the data points, even though the feature space has only 7 dimensions. The computation time to obtain the rule-list is also in the order of few hours for the 7-feature model, while we provide the explanation report for an input with 88 features in a few seconds.

Google provides an interactive tool called What-If [45] which enables practitioner to probe the behavior of their models by providing hypothetical inputs and observing how the model output changes. For each labeled data point in the dataset, the tool reports the closest data point to it in another class, but it is not equipped to investigate the decision boundaries of the models.

Koh and Liang [16] and Koh et al. [17] have used influence functions to reveal the importance of individual training data in forming the trained model, but their method cannot be used to explain outputs of the models or to investigate the decision boundaries.

There are studies in deep learning that consider the decision boundaries from other perspectives. For example, Elsayed et al. [7] and Jiang et al. [14] correctly define the decision boundaries, then use first-order Taylor series approximation to estimate the distance to them for individual inputs, and study the distance in relation to generalization error in deep learning. However, those approximation methods are shown to be unreliable for nonlinear models [49]. Methods to generate adversarial inputs, for example [8, 13, 25], apply small perturbations to an input until its classification changes, but those methods do not seek the closest point on the decision boundaries, and therefore cannot find the least changes required to change the model's output. Most recent methods for computing adversarial inputs, such as Ilyas et al. [12] and Tsipras et al. [41], also do not seek points on or near the decision boundaries.

Recently, Lundberg et al. [23] studied interpretation of tree models, but their method is only applicable to trees. Another approach by Lundberg and Lee [22] borrows Shapely regression values from the linear regression literature in order to assign importance values to features of a model. This tends to work better than LIME in identifying feature importance, but the provided information is limited in scope and not adequate to thoroughly audit a model and investigate its decision boundaries.

3 Defining and computing flip points

In this section, we first define the model to be a function, then review the work on flip points in [50] and extend that work by defining the constrained flip points.

3.1 Defining a trained classification model as a mathematical function

As mentioned earlier, we view the trained model as a mathematical function that maps inputs to outputs. Consider a model $\mathcal{N}$ that has n continuous outputs corresponding to n classes. The output of the model is a vector

$$\boldsymbol{z} = \mathcal{N}(\boldsymbol{x}) \tag{1}$$

where each element z_i corresponds to a class. For convenience and as is customary in practice, we assume that elements of $\boldsymbol{z}$ are normalized to sum to 1 (e.g., by softmax). For any given input, the model first computes the $\boldsymbol{z}$, and then its classification is defined by the fuction

$$\mathcal{C}(\boldsymbol{z}) = \{i : z_i = \max_k z_k\}, \tag{2}$$

which identifies the maximal components of $\boldsymbol{z}$ and thus defines the assigned class(es).[5] When the largest value of $\boldsymbol{z}$ is unique, the classification of $\boldsymbol{x}$ by the model is that class i. But when there are ties, we consider the output of the model to be a *flip* between the corresponding classes. Such flip points form the decision boundaries of $\mathcal{N}$ and partition its domain. In later sections, we will discuss and formulate optimization problems to find such points, but first, let's consider the inner workings of $\mathcal{N}$.

Our methods can be applied to general models, but our computational results use neural networks. In this case, the trained model $\mathcal{N}$, or in other words our deep learning function, is composed of neurons organized in various layers. A given input gets processed through the layers of the model until it reaches the output layer, a.k.a. forward propagation. The computational process at a single layer of a neural network typically involves multiplying the layer's input by a weight matrix, then adding a bias vector to that, and finally, applying an activation function in order to produce the output of the layer and pass it on to the next layer. Additional layers may also exist, such as convolutional layers (i.e., linear transformation of data with a stencil). Moreover, there are many options for the activation function of neurons that are interchangeably used both in practice and in the literature, e.g., ReLU, error function, sigmoid, hyperbolic tangent, etc.

Regardless of what specific architecture is used in the model and what activation function is used for the neurons, we usually know how to compute the gradient of the output of $\mathcal{N}$ with respect to its input. In fact, for standard neural network architectures and standard activation functions, the gradients of the output of $\mathcal{N}$ can be written explicitly based on the chain rule decomposition of operations performed in the layers of the network, a.k.a. back propagation.[6]

[5] As an example, consider that $\boldsymbol{x}$ contains census information about individuals, and each element of $\boldsymbol{z}$ corresponds to one income bracket/category (e.g., income below poverty line, low income, high income, etc.), i.e., the model wants to predict the income category of individuals based on their census information.

[6] Numerical packages have built-in functions that return the gradients for any model with a typical architecture and with any of the common activation functions.

We are interested in solving optimization problems about the trained model, $\mathcal{N}$. Hence, we expect the $\mathcal{N}$ to appear as a function in the objective function and/or the constraints of the optimization problem. If we can compute the gradient of the output of $\mathcal{N}$ w.r.t. its input, our optimization problems involving $\mathcal{N}$ are somewhat easier.

Difficulties may arise in solving any optimization problem and there is no exception for the problems we consider in this paper. Specifically, the issue of vanishing and exploding gradients is a common problem that one may encounter when dealing with the gradients of neural networks [2, 11]. We have studied these issues in our previous work [50], and will briefly review them in Section 3.6.

It is worthwhile to note that the models we want to audit/debug are already trained. Whether $\mathcal{N}$ has a good architecture or bad architecture and whether it is trained in the best possible way or not, from this paper's perspective, $\mathcal{N}$ is a function, and we propose to audit its behavior on the individual-level and on the group-level. In the next section, we discuss the optimization problem that can achieve this goal, and build other pieces of our auditing procedure afterwards.

3.2 Defining flip points

We mentioned earlier that a trained classification model is defined by its decision boundaries. We refer to the points on the decision boundaries as *flip points*. Particularly, for a given input $\boldsymbol{x}$, we are interested in the closest flip point to $\boldsymbol{x}$, i.e., the least changes in $\boldsymbol{x}$ that change the decision of the model to another class.

A point on the decision boundary of the model between classes i and j, denoted by $\hat{\boldsymbol{x}}^c_{i,j}$, should satisfy these constraints:

$$\boldsymbol{z} = \mathcal{N}(\hat{\boldsymbol{x}}^c_{i,j}), \tag{3}$$

$$z_i = z_j. \tag{4}$$

$$z_i \geq z_k, \forall k \notin \{i, j\}, \tag{5}$$

$$\hat{\boldsymbol{x}}^c_{i,j} \text{ is within the domain of } \mathcal{N}. \tag{6}$$

This means that the model $\mathcal{N}$ generates equal scores for classes i and j and that no score for another class exceeds those scores. This definition is strict in the sense that it requires the point to be at the interface of classes i and j. [7]

[7] In theory, if $n > 2$, there may fail to be a point satisfying (3)-(6), which means that partitions of class i and class j have no common boundary. We have not encountered this in practice and previous studies on decision boundaries have not encountered it either [9], but it would mean that there is little chance of confusion between these classes, since any

In binary classification ($n = 2$), the constraints (4)-(5) can be reduced to

$$z_1 = 1/2, \tag{7}$$

which forces $z_2 = 1/2$, since they add to 1.

Now that we have formulated the conditions that define the points on the decision boundaries, we can seek the $\hat{\boldsymbol{x}}^c_{i,j}$ that is closest to a given input $\boldsymbol{x}$. For that, we define our objective function as

$$\min_{\hat{\boldsymbol{x}}^c_{i,j}} \|\hat{\boldsymbol{x}}^c_{i,j} - \boldsymbol{x}\|, \tag{8}$$

where $\|.\|$ is a norm appropriate to the data.

Minimizing (8) subject to constraints (3)-(6) define our optimization problem for finding the closest flip point to $\boldsymbol{x}$ between classes i and j. Constraint (6) often translates into upper and lower bounds on the continuous features such as age, or integer constraints on discrete features such as race.

Specific problems might require additional constraints. For example, we might want to restrict the search to a specific subspace of the domain, fix some of the variables, etc. It is possible that the solution $\hat{\boldsymbol{x}}^c_{i,j}$ is not unique, but the minimal distance is always unique.

In theory, another special case that can arise with $n > 2$ is if for some $\boldsymbol{x}$ (which belongs to some class i according to $\mathcal{N}$), our optimization problem yields some $\hat{\boldsymbol{x}}^c_{i,j}$ as the flip point between class i and some class j such that there exists no path (linear or curved) between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}^c_{i,j}$ that remains inside a single partition of $\mathcal{N}$. This might imply that the specific partition, p_i, that contains $\boldsymbol{x}$, does not have a common boundary with partitions of class j, but there is a separate partition p'_i of class i that has a common boundary with class j. Alternatively, it may imply that p_i has a common boundary with class j, but there is some other partition p'_i separated from p_i, and the interface of p'_i with class j has points closer to $\boldsymbol{x}$ than the interface of p_i with class j. To make sure that $\hat{\boldsymbol{x}}^c_{i,j}$ is at the interface of same partition that contains $\boldsymbol{x}$, we can consider adding another constraint

$$\exists \pi, \pi : [\boldsymbol{x}, \hat{\boldsymbol{x}}^c_{i,j}) \mapsto \text{single partition in the domain of } \mathcal{N}, \tag{9}$$

where π is any path between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}^c_{i,j}$. In our experiments on a wide range of models, constraint (9) has always been automatically satisfied with π being a direct line connecting $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}^c_{i,j}$. In fact, studies on the geometry of decision boundaries report that there are paths connecting all samples of the same class, revealing that one united region defines the partition for each class [9, 49], naturally satisfying the constraint (9).

It is also possible that a flip point is at the interface of multiple classes, let's say, as an example, a flip point between 4 classes defined by $\phi = \{i, j, l, m\}$. We can denote such flip point by $\hat{\boldsymbol{x}}^c_{i,j,l,m}$ satisfying

path from one to the other passes through yet another region. This would be a useful piece of information for the auditor to know about the model. In such instance, one can seek a non-strict definition of flip point by dropping the constraint (5).

$$\boldsymbol{z} = \mathcal{N}(\hat{\boldsymbol{x}}^c_{i,j,l,m}),$$
$$z_i = z_j = z_l = z_m,$$
$$z_i > z_k, \forall k \notin \phi.$$

3.3 How does our formulation compare with the counterfactual formulation in literature?

We now consider the formulations present in the literature for computing counterfactuals, and explain how they differ from our formulation. Let's consider the formulation used by Wachter et al. [44] and several others. Their equation 2 defines the counterfactual $\mathbf{c}$ corresponding to input $\boldsymbol{x}$ as

$$\arg\min_{\mathbf{c}} \max_{\lambda} \lambda(\mathcal{N}(c) - l)^2 + d(\boldsymbol{x}, \mathbf{c}), \tag{10}$$

where l is the label for the target class, λ is a scalar to control the first term of objective function, and $d(.)$ is a distance function which they use L_1 norm weighted by the inverse median absolute deviation. This formulation resembles a dual form, however, it is incomplete and mathematically ill defined and its solution is infinity. That is the reason why Wachter et al. [44] enumerate all the input space and plug them into the model in order to find the closest counterfactual. But this approach of enumerating is not practical in most cases.

Mothilal et al. [26] modify the equation (10) to this form:

$$\arg\min_{\mathbf{c}} \mathit{hinge}(\mathcal{N}(c), l) + |\boldsymbol{x} - \mathbf{c}|, \tag{11}$$

where $\mathit{hinge}(.)$ is the hinge loss function. They recommend using $l = 0.5$, but they do not note that $l = 0.5$ is only a special case applicable for binary classification.

Even in the binary classification case and using $l = 0.5$, the formulation defined by (11) is ill-defined and its solution is not necessarily a point on the decision boundary. Mothilal et al. [26] report that sometimes they cannot find any point on the decision boundaries without realizing that they are solving an ill-defined optimization problem. In fact, it is surprising to naturally arrive at a decision boundary point by merely solving this optimization problem. To gain further insights, compare (11) with the optimization problem we defined in the previous section.

In summary, any auditing method that relies on solving an ill-defined optimization problem can mislead the auditor. First, one might not be able to solve the optimization problem at all, regardless of what optimization algorithm they use. This is evidently the reason why Wachter et al. [44] enumerate the input space, and also the reason why Mothilal et al. [26] report that for some inputs they fail to solve the optimization problem. Second, even if one

manages to find some feasible solution, the solution to the ill-defined optimization problem is most likely not the closest counterfactual, and therefore, it might mislead the auditor about the behavior of the model. It is important for any auditing procedure to rely upon a well-defined formulation.

Now that we have established our formulation and compared it with the popular methods in literature, in the next two sections, we will consider constrained flip points. Later in Section 3.6, we will review the optimization algorithm to solve the problem.

3.4 Constrained flip points

The closest flip point is the closest point on the decision boundary of the model to a particular input. In certain cases, we may be interested in the influence of a particular feature or a subset of features. In such cases, we find the closest flip point in the subspace of the domain representing only the features of interest.

In order to find such flip points, we have to restrict the domain by adding more constraints to the optimization problem of Section 3.2, allowing only a subset of features to vary. This leads us to defining *constrained* flip point: a point that is on the decision boundaries of the $\mathcal{N}$ and is closest to a given input in certain sub-spaces of the domain. Note that such additional constraints usually imply having fewer variables in our optimization problems, making it easier to solve.

3.5 Two notes on defining flip points

Knowing the dependencies among the features can help in choosing meaningful subsets of features for computing constrained flip points. For example, "income" and "net worth" may be correlated in a dataset. If we choose to vary a subset of features that contains "income" while holding "net worth" constant, the constrained flip point might not be very meaningful.

So for many reasons, it can be desirable to identify the dependencies among the features in a dataset. In our computational examples, we do this using the pivoted QR decomposition [10, Chap. 5] of a data matrix $\boldsymbol{D}$ whose rows are the training data points and whose columns are features. This decomposition reorders the columns, pushing linearly dependent columns (redundant features) to the right and forming

$$\boldsymbol{DP} = \boldsymbol{QR},$$

where $\boldsymbol{P}$ is the permutation matrix, $\boldsymbol{Q}$ has orthogonal columns, and $\boldsymbol{R}$ is zero below its main diagonal. The degree of independence of the features can be determined by measuring the matrix condition number of leading principal submatrices of $\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ 0 & \boldsymbol{R}_{22} \end{pmatrix}$, or by taking the matrix norm of trailing sets of columns, i.e., the norm of $\boldsymbol{R}_{22}$.The numerical rank of $\boldsymbol{D}$ is the dimension of

the largest leading principal submatrix of $\boldsymbol{R}$ with a sufficiently small condition number.

Alternatively, the singular value decomposition (SVD) of $\boldsymbol{D}$ can be used in a similar way [10, Chap. 2]. In this case, the numerical rank is the number of sufficiently large singular values. The SVD will identify principal components (i.e., linear combinations of features in decreasing order of importance), and unimportant ones can be omitted. The most significant combinations of features can be used as training inputs, instead of the original features.

The underlying metric of these matrix decompositions is the Euclidean norm, so they are most easily justified for continuous features measured on a single scale, for example, pixel values in an image. For disparate features, the scale factors used by practitioners to define an appropriate norm for the optimization problem in Section 3.2 can be used to renormalize features before forming $\boldsymbol{D}$. Leaving the choice of scale factor to practitioners is suggested by Spangher et al. [40] and Wachter et al. [44], too.

3.6 Notes about computing flip points

Our optimization problem (8), subject to (3)-(6) can be solved by off-the-shelf or specialized algorithms that determine local minimizers for non-convex problems. Here, we briefly review the issues that may arise in solving the optimization problem and also the algorithm we have developed in our previous work [50].

The main challenges in solving our optimization problem can be summarized as the following:

1. High non-linearity in the constraints that involve $\mathcal{N}$
2. Non-convexity
3. High-dimensionality of the domain in most cases
4. A combination of discrete and continuous variables (as in the examples we consider in this paper)
5. Vanishing and exploding gradients of $\mathcal{N}$

There is a rich literature in numerical optimization for overcoming issues 1 through 4. The 5th issue, however, is a phenomenon, explainable by the flow of gradients in neural networks. When computing the gradients of a neural network, some elements in the gradient matrix may tend to reach zero (vanish), while other elements may become very large in absolute value (explode). This may eventually lead to a badly scaled and uninformative gradient matrix, and by extension, to an ill-conditioned optimization problem.

Another issue that may arise is satisfying the constraints of the optimization problem. Sometimes finding any feasible solution may be challenging. Despite all these challenges, an off-the-shelf algorithm from a high-quality package such as NLOPT [15] might be able to find a solution to this optimization problem.

To overcome these difficulties, in previous work [50], we developed a homotopy algorithm that transforms the $\mathcal{N}$ and then gradually transforms it back to its original form while tracing a path of solutions, using a local (off-the-shelf) solver. The transformation of the model ensures that the absolute value of the gradients of neurons are bounded away from zero and are also upper bounded by 1. This ensures that the gradients of the transformed model are well-scaled and informative. Moreover, the homotopy transformation changes the bias parameters of the last layer of the model (by solving a convex optimization problem) to ensure that the homotopy path starts with a known solution. We have provided specific implementations for models with ReLU and erf activation functions, but the concept of tuning the activation functions to bound the gradients is applicable to other activation functions, too.

Homotopy optimization algorithms are studied in the past few decades in the mathematical optimization literature, usually for problems that are hard to solve [6, 28]. The specific homotopy is chosen to overcome the undesirable aspects of the problem and to facilitate a path towards finding the solution for the original problem.

Our homotopy algorithm in [50] makes us more likely to find a feasible solution for our optimization problem, and as we have demonstrated in [48], the homotopy algorithm may find solutions that are closer to the original input. However, we note that there is no guarantee that our homotopy algorithm will find a feasible solution, nor there is a guarantee that it will perform better than other algorithms, as discussed in detail in [50].

Moreover, we note that our optimization problem is non-convex, and there is no guarantee that the closest flip point we find is actually the global solution of the optimization problem. As is the case with solving all non-convex problems, the proper mathematical approach is to utilize one or a few global optimization algorithms and use the best feasible solution that we find as our best estimate for the global solution of the problem. It is important to realize that studies in the literature have usually settled for first-order Taylor series approximation, or taking small gradient based steps until crossing a decision boundary. In [49], we showed that those approximation methods may be unreliable.

For the local solver used in the homotopy algorithm, overall, we have had better success in using interior-point algorithms, while in many cases, trust-region methods can be as effective. To deal with the combination of discrete and continuous features, we initially relax the discrete space into a continuous space, then gradually enforce the discrete requirement by adding a penalty term to the objective function.

Finally, we note that an auditor may have limited access to some models, for example, they may only be able to feed inputs to a model and receive the outputs without having access to the gradients. In such cases, one option is to use a finite difference method to estimate the gradient of the output of the model w.r.t. its input. Another option may be to use gradient-free optimization algorithms. The numerical optimization literature suggests that gradient-

based algorithms are usually more capable than gradient-free algorithms, and estimating the gradients is usually worth the computational cost.

3.7 Computational cost

For a neural network, the cost of each iteration in determining a flip point is dominated by the cost of computing the gradient of the output of the model with respect to its input. That cost is equivalent to the cost of computing the output of the model for that input.

So, assuming that we want to audit a particular model that is already in use on a computer, that computer would be able to compute the flip point and the explanation report as well. If the auditor wants the closest flip points for an entire dataset, they can be computed in parallel. For the examples we provide in this paper, computing a closest flip point just takes a few seconds on a 2017 Macbook.

In previous work, we have experimented with image classification models [50] with thousands of variables and found that solution times were still reasonable.

4 Using flip points to explain, audit and debug models

4.1 Individual-level auditing: Providing explanations and feedback to users of a model

To generate a report like that in Figure 1, we need to compute flip points and constrained flip points in order to determine the *smallest* changes in the features that change the model's output. Algorithm 1 summarizes the use of constrained flip points in generating such a report, giving a user precise information on how individual features and combinations of features influenced the model's recommendation for a given input.

Algorithm 1 Using constrained flip points to generate an explanation for a model's output for a specific input $\boldsymbol{x}$

Given: a trained model $\mathcal{N}$, a specific input $\boldsymbol{x}$, and desired subsets of features to be investigated
Produce: an explanation report, giving various insights about the model's output for $\boldsymbol{x}$

1: Compute the closest flip point to $\boldsymbol{x}$
2: Compute constrained flip points for $\boldsymbol{x}$, allowing one feature to change at a time
3: Group the features that have the same measurement scale and compute the constrained flip points for $\boldsymbol{x}$ in subspaces defined by each feature group
4: Compute the constrained flip point for $\boldsymbol{x}$ allowing any desired subset of features to change
5: Generate an explanation report based on the computed flip point and constrained flip points

4.2 Group-level auditing: Studying the behavior of a model towards groups of individuals

It is important to audit and explain the behavior of models, not only on the individual-level, but also towards groups. Groups of interest can be an entire dataset or specific subsets within it, such as people with certain age, gender, education, etc. The information obtained from the group-level analysis can reveal systematic traits or biases in a model's behavior. It can also reveal the role of individual features or combinations of features on the overall behavior of model.

Algorithm 2 presents some of the ways that flip points can yield insight into these matters. By computing the closest flip points for a group of individuals, we obtain the vectors of directions to the decision boundary for them. We call these directions *flip directions*. Using pivoted QR decomposition or principal component analysis (PCA) on the vectors of directions, we can identify important patterns and traits in a model's decision making for the group of individuals under study.

For example, consider auditing a cancer prediction model for a group of individuals with cancerous tumors. After computing the flip directions, we can study the patterns of change for that population, e.g., which features have changed most significantly and in which direction.

We can also study the effect of specific features on a model's decision making for specific groups. For this type of analysis, we compute constrained flip points for the individuals in the group, allowing only the feature(s) of interest to change. We then study patterns in the directions of change. For example, when auditing a model trained to evaluate loan applications, we might examine the effect of age for people who have been denied. We can compute constrained flip points for those individuals, allowing only the feature of age to change, and then study the patterns in flip directions, i.e., in which direction "age" should change and to what extent in order to change the decisions for that population.

We might also want to examine the effect of gender for the same loan application model. To do this, we pair each data point with an identical one but of opposite gender. We compute flip points for all of the inputs and look for patterns: For the paired points whose classification did not change, did the mean/median distance to the decision boundary change significantly? For the points whose classification changed, do the directions to the boundary have any commonalities, as revealed by pivoted QR or principal component analysis (PCA)?

4.3 Debugging a model

If we determine that the model's behavior is undesirable for a particular set of inputs, we would like to alter the decision boundaries to change that behavior. For example, when there is bias towards a certain feature, it usually

Algorithm 2 Auditing a model's behavior on training or testing data

Given: a trained model $\mathcal{N}$ and a data matrix $\boldsymbol{D}$
Produce: various insights into the behavior of the model

1: Compute the closest (or constrained) flip points for all the data in $\boldsymbol{D}$, forming a matrix $\boldsymbol{B}$.
2: For correctly classified points (and then again for incorrectly classified ones), form $\boldsymbol{F} = \boldsymbol{B} - \boldsymbol{D}$, the matrix of directions from data points to flip points
3: Perform pivoted QR on $\boldsymbol{F}$ to identify features that are most and least influential in flipping the decisions of the model.
4: If $\boldsymbol{F}$ is close to rank deficient, then the set of directions to the decision boundary is of lower dimension than the number of features and it would be insightful to investigate the source of rank deficiency, i.e., zero columns and/or linearly dependent columns and their corresponding features.
5: Compute the principal components of $\boldsymbol{F}$ to identify commonalities among the directions to the boundary from the training points.
6: Study the frequency of change between points and their flip points for each feature to gain insight about influence of features. Some features may change rarely among the population while some features may change frequently, indicating traits about the model attitude towards groups.
7: For a (binary) feature that should not affect the output classification, consider the dataset $\hat{\boldsymbol{D}}$ that has the opposite value for that feature. Compute the resulting classifications. For points whose classification did not change, compute the mean change in distance to the boundary; ideally, this will be small. For points whose classification changed, pivoted QR or PCA analysis on the direction matrix will identify possible sources of the model's rationale.

means data points are close to decision boundaries in that feature dimension. By computing constrained flip points in that dimension, adding them to the training set with the same label, and retraining, we can push the decision boundaries away from the inputs in that dimension. This tends to change the behavior of models, as we show in our numerical results.

From the practical point of view, the additional training points are synthetic data that we do not have in the dataset, but having them would be desirable. For example, if we have a dataset that is biased against a certain race, the synthetic data may be considered samples that we wish to have in the data in order to remove or alleviate that bias. Generating such synthetic data by way of flip points, not only is effective in improving the model (because it directly relates to the location of decision boundaries), it can also be an inexpensive and fast alternative to gathering unbiased data from the real world.

Moving the decision boundaries away from the training data also tends to improve the generalization of deep learning models as reported by Elsayed et al. [7] and Yousefzadeh [48]. It is also possible to create flip points and teach them to the model with a flip label (i.e., $z_1 = z_2 = 1/2$), in order to define a decision boundary in certain locations. In our numerical results, we will explore the benefits of generating and using synthetic data.

5 Results

Here, we demonstrate our techniques for explaining, auditing, and debugging deep learning models on three different datasets with societal themes. We use three software packages, NLopt [15], IPOPT [42], and the Optimization Toolbox of MATLAB, as well as our own custom-designed homotopy algorithm [50], to solve the optimization problems. The algorithms almost always converge to the same point. The variety and abundance of global and local optimization algorithms in the above optimization packages give us confidence that we have indeed usually found the closest flip point. Our code for computing flip points is available online as explained in Appendix B, and details of the models are given in Appendix C. The datasets are public; see footnotes 1,2,3.

For the two first examples, the FICO challenge and the Credit dataset, we compare our results with two recent papers that have used those datasets. To make the comparison fair and easy, for each dataset we make the same choices about the data (such as cross validation, portion of testing set, etc.) as each of those papers.

5.1 FICO Explainable ML Challenge

This dataset has 10,459 observations with 23 features, and each data point is labeled as "Good" or "Bad" risk. We randomly pick 20% of the data as the testing set and keep the rest as the training set. We regard all features as continuous, since even "months" can be measured that way. The description of features is provided in Appendix A.

5.1.1 Eliminating redundant features

The condition number of the matrix formed from the training set is 653. Pivoted QR factorization finds that features "MSinceMostRecentTradeOpen", "NumTrades90Ever2DerogPubRec", and "NumInqLast6Mexcl7days" are the most dependent columns; discarding them leads to a training set with condition number 59. Using the data with 20 features, we train a neural network with 5 layers, achieving 72.90% accuracy on the testing set. A similar network trained with all 23 features achieved 70.79% accuracy, confirming the effectiveness of our decision to discard the three features. Rudin et al. [36] had reported the positive effect of eliminating redundant features, too.

5.1.2 Individual-level explanations

As an example, consider the first datapoint, corresponding to a person with "Bad" risk performance. The feature values for this data point are provided in Appendix A. The closest (unconstrained) flip point is virtually identical to the data point except in five features, shown in Table 1.

Next, we allow only a subset of the features to change and compute constrained flip points. We explore the following subspaces:

Table 1: Difference in features for data point # 1 in the FICO dataset and its closest flip point.

Feature	Input #1	Closest flip point (relaxed)	Closest flip point (integer)
AverageMInFile	84	105.6	111.2
NumSatisfactoryTrades	20	24.1	24
MSinceMostRecentDelq	2	0.6	0
NumTradesOpeninLast12M	1	1.7	2
NetFractionRevolvingBurden	33	19.4	8.5

1. Only one feature is allowed to change at a time. None of the 20 features is individually capable of flipping the decision of the model.
2. Pairs of features are allowed to change at a time. Only a few of the pairs (29 out of 190) can flip the output. 13 of these pairs involve the feature "MSinceMostRecentInqexcl7days" as partially reflected in the explanation report of Figure 2.
3. Combinations of features that share the same measurement scale are allowed to change at a time. We have five distinct groups: features that are measured in "percentage", "number of months", "number of trades", "delinquency measure", and "net fraction burden". The last two feature groups are not capable of flipping the prediction of the model by themselves.

The explanation summary report resulting from these computations is shown in Figure 2. The top two sections show the results of computing constrained flip points, first, points where no constrained flip point exists and the label does not change, and then points with different label. The bottom section displays the unconstrained flip point. This shows that the output of a deep learning model can be explained clearly and accurately to the user to any desired level of detail. The answer to other specific questions can also be found easily by modifying the optimization problem.

We note that the time it takes to find each flip point is only a few milliseconds using a 2017 MacBook, hence this report can be generated in real-time.

5.1.3 **Group-level explanations**

Using pivoted QR on the matrix of directions between data points labeled "Bad" and their flip points, we find that, individually, the three most influential features are "AverageMInFile", "NumInqLast6M", and "NumBank2 NatlTradesWHighUtilization". Similarly, for the directions that flip a "Good" to a "Bad", the three most influential features are "AverageMInFile", "NumInqLast6M", and "NetFractionRevolvingBurden". In both cases, "ExternalRiskEstimate" has no influence.

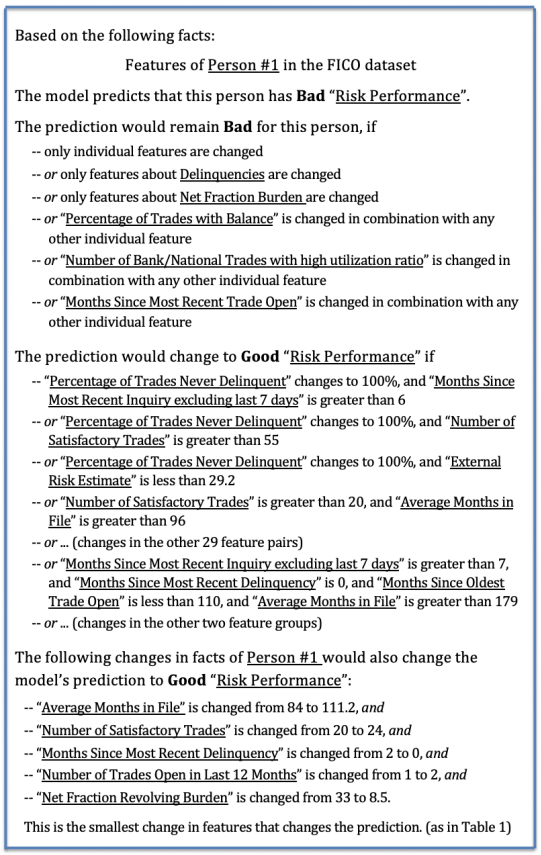

Fig. 2: A sample explanation report for data point #1 in the FICO dataset, classified by a deep learning model.

We perform PCA analysis on the subset of directions that flip a "Bad" to "Good" risk performance. The first principal component reveals that, for this model, the most prominent features with positive impact are "PercentTradesNeverDelq" and "PercentTradesWBalance", while the features with most negative impact are "MaxDelqEver" and "MSinceMostRecentDelq". These conclusions are similar to the influential features reported by [3], however, our method gives more detailed insights, since it includes an individual-level explanation report and also analysis of the group effects.

5.1.4 Effect of redundant variables on adversarial vulnerabilities

Interestingly, for the model trained on all 23 features, the three most significant individual features in flipping its decisions are "MSinceMostRecentTradeOpen", "NumTrades90Ever2DerogPubRec" and "NumInqLast6Mexcl7days", exactly the three dependent features that we discarded for the reduced model.

Thus, the decision of the trained model is more susceptible to changes in the dependent features, compared to changes in the independent features.

This reveals an important vulnerability of machine learning models regarding their training sets. For this dataset, when redundant features are included, they become the most influential features in flipping the decisions of the model, making the model vulnerable to adversaries, because they would be able to fool the model easily by making arbitrary changes to the redundant features.

5.1.5 **Auditing the model using flip directions**

Figure 3 shows the directions of change to move from the inputs to the closest flip points for features "NumInqLast6M" and "NetFractionRevolvingBurden", which are the most influential features given by the pivoted QR algorithm. Even though flip points are unconstrained, directions of change for these two features are distinctly clustered for flipping a "Bad" label to "Good" and vice versa.

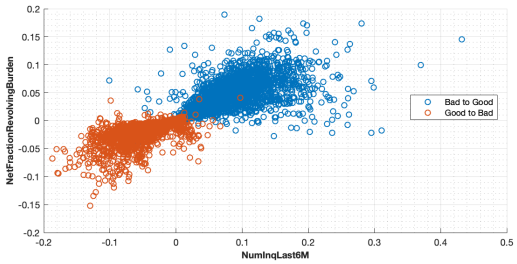

Fig. 3: Directions between the inputs and their closest flip point for two influential features. Points are distinctly clustered based on the direction of the flip.

Furthermore, Figure 4 shows the directions in coordinates of the first two principal components. We can see that the flip directions are clearly clustered into two convex cones, exactly in opposite directions. Also, we see that misclassified inputs are relatively close to their flip points while correct predictions can be close or far.

5.1.6 **Comparison**

The interpretable model developed by Chen et al. [3] reports the most influential features which are similar to our findings above, e.g., "PercentTradesNeverDelq" and "AverageMInFile". However, their model is inherently interpretable, and their auditing method is not applicable to deep learning models. They also do not provide an explanation report on the individual-level, like the one we provided in Figure 2.

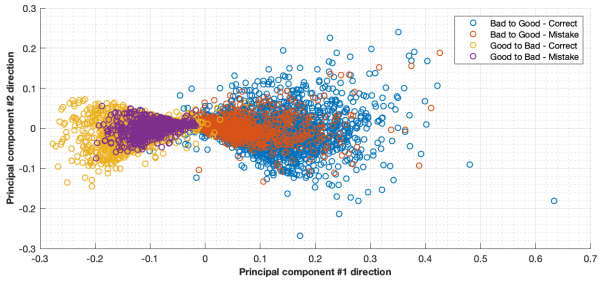

Fig. 4: Change between the inputs and their unconstrained flip points in the first two principal components. Directions are clustered into two convex cones, exactly in opposite directions.

We note that our goal, here, is to show how a deep learning model used for this application can be audited. We do not necessarily advocate for use of deep learning models over other models.

5.2 Default of credit card clients

This dataset [47] from the UCI Machine Learning Repository has 30,000 observations, 24 features, and a binary label predicting whether or not the person will default on the next payment.

We binarize the categorical variables "Gender", "Education", and "Marital status"; the categories that are active for a data point have binary value of 1 in their corresponding features, while the other categories are set to zero. When searching for a flip point, we allow exactly one binary feature to be equal to 1 for each of the categorical variables. The condition number of the training set is 129 which implies linear independence of features. Using a 10-fold cross validation on the data, we train a neural network with 5 layers (details in Appendix C), to achieve accuracy of 81.8% on the testing set, slightly higher than the accuracy of around 80.6% reported by Spangher et al. [40]. When calculating the closest flip points, we require the categorical variables to remain discrete.

5.2.1 Individual-level explanations

We consider the data point #1 in this dataset which is classified as "default", and compose the explanation report shown in Figure 5. When we examine the effect of features, we see that any of 4 features can flip the prediction of the model, individually.

When examining this report for input #1, we find some flaws in the model. For example, in order to flip the prediction of the model to non-default, one option is to reduce the amount of the current bill to -$2,310,000, while reducing the bill to any number larger than that would not flip the prediction.

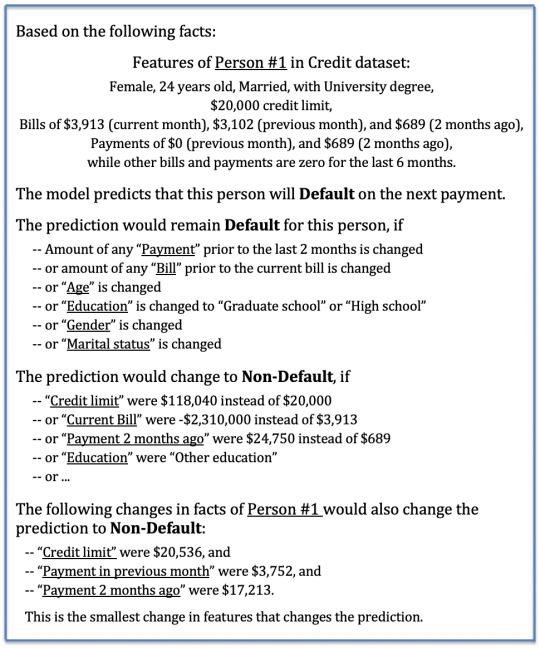

Fig. 5: A sample explanation report for data point #1 in the Credit dataset, predicted to default on the next payment. The deep learning predicts the labels for the testing data well. But, what it takes to change the prediction of the model sometimes does not seem rational.

Requiring any negative balance on the bill is irrational, because as long as the bill is zero, there would be no chance of default. In fact, one would expect the prediction of non-default if the current bill is changed to zero, for any datapoint. But, the training set does not include such examples, and clearly, our model has not learned such an axiom. Requiring the large payment of $24,750 (for 2 months ago) in order to flip the prediction seems questionable, too, considering that the current bill is $3,913.

Therefore, despite the model's good accuracy on the testing data, the explanation for its prediction reveals flaws in its behavior for data point #1. These flaws would not have been noticed without investigating the decision boundaries. Fortunately, because of our auditing, we know that the model needs to be improved before it is deployed. Although this dataset has been the subject of numerous studies, this limitation (of not having samples with zero balance) was not detected and reported previously in the literature.

Clearly, one can compare the flip points with the training set and conclude that certain flip points are out-of-distribution, revealing the limitation of what the model has learned from the data. On that ground, one can reject certain flip points, and by extension, decisions of a model. We believe this can become a standard procedure when auditing deep learning models.

5.2.2 Group-level auditing using flip points

Examining the flip points for the training data reveals model characteristics that should be understood by the users. Here is one example.

Gender does not have much influence in the decisions of the model, as only about 0.5% of inputs have a different gender than their flip points. Hence, gender is not an influential feature for this model. This kind of analysis can be performed for all the features, in more detail.

5.2.3 Group-level auditing using flip directions

We perform pivoted QR decomposition on the directions to the closest flip points. The results show that "BILL-AMT3"[8] and "BILL-AMT5" are the most influential features, and "Age" has the least influence in changing the predictions. In fact, there is no significant change between the age of any of the inputs and their closest flip points.

5.2.4 Debugging the model using flip points

In both our training and testing sets, about 52% of individuals have age less than 35. Following Spangher et al. [40], we remove 70% of the young individuals from the training set, so that they are under-sampled. We keep the testing set as before and obtain 80.83% accuracy on the original testing set. We observe that now, "Age" is the 3^{rd} most influential feature in flipping its decisions. Moreover, PCA analysis shows that lower Age has a negative impact on the "no default" prediction and vice versa.

We consider all the data points in the training set labelled as "default" that have closest flip point with older age, and all the points labelled "no default" that have closest flip point with younger age. We add all those flip points to the training set, with the same label as their corresponding data point, and train a new model. Now Age has become the 11^{th} most influential feature and it is no longer significant in the first principal component of the flip directions; hence, the bias against Age has been reduced. Also, testing accuracy slightly increases to 80.9%.

Adding synthetic data to the training set has great potential to change the behavior of a model, but we cannot rule out unintended consequences. By investigating the influential features and PCA analysis, we see that the model has been altered only with respect to the Age feature, and the overall behavior of model has not changed.

5.2.5 Comparison

Spangher et al. [40] used a logistic regression model for this dataset, achieving 80.6% accuracy on testing, less than our 81.8% accuracy. Their method

[8] "BILL-AMTx" stands for the Amount of Bill in $, x month(s) ago.

for computing flip points is limited to linear models and not applicable to deep learning. They also do not provide an explanation report like the one in Figure 5.

They have reported that under-sampling young individuals from the training set makes their model biased towards young age, similar to ours. However, they do not use flip points to reduce the bias, which we successfully did.

5.3 Adult Income dataset

The Adult dataset from the UCI Machine Learning Repository [5] has a combination of discrete and continuous variables. Each of the 32,561 data points in the training set and 16,281 in the testing set are labeled, indicating whether the individual's income is greater than 50K annually. There are 6 continuous variables including Age, Years of education, Capital-gain, Capital-loss, and Hours-per-week of work. We binarize the discrete variables: Work-class, Marital status, Occupation, Relationship, Race, Gender, and Native country. Our trained model considers 88 features and achieves accuracy 86.08% on the testing sets, comparable to best results in the literature [5]. Our aim here is to show how a trained model can be audited.

5.3.1 Individual-level auditing

As an example, consider the first data point in the testing set, corresponding to a 25-year-old Black Male, with 11th grade education and native country of United States, working 40 hours per week in the Private sector as Machine-operator-inspector and income "$\leq$ 50K", correctly classified by the model. He has never married and has a child/children.

We compute the closest flip point for this individual, allowing all the features to change. Table 2 shows the features that have changed for this person in order to flip the model's classification for him to the high income bracket. Other features such as gender, race, work-class have not changed and are not shown in the table. Directions of change in the features are generally sensible: e.g., working more hours, getting a higher education, working in the Tech-sector, and being older generally have a direct relationship with higher income. Being married instead of being a single parent is also known to have a relationship with higher income.

We further observe that none of the features individually can flip the classification, but certain constrained flip points can provide additional insights about the behavior of the model.

Let's consider the effect of race. The softmax score for this individual is 0.9989 for income "$\leq$ 50K". Changing the race does not affect the softmax score more than 0.0007. This observation about softmax score might lead one to believe that the model is neutral about race, at least for this individual. However, that would not be completely accurate in all circumstances, as we will explain. If we keep all features of this individual the same and only change

Table 2: Difference in features for Adult dataset testing point #1 and its closest flip point.

Data	Input #1 in testing set	Closest flip point
Age	25	30.3
Years of education	7 (11th grade)	8 (12th grade)
Marital status	Never-married	Married-ArmedForcesSpouse
Relationship	Own-child	Husband
Occupation	Machine-operation-inspection	Tech-support
Hours-per-week of work	40	41.8

his race to Asian, the closest flip point for him would be the same as before, except for Age of 29.9 and Hours-per-week of 42.3. The differences in flip points for the Black and Asian are not large enough to draw a conclusion.

This result is novel and not reported in the literature. It shows that certain behaviors and biases of a model towards a certain feature may be well hidden in other subspaces, not recognizable by merely changing the variable of interest. In order to audit the models, we actually need to examine the exact location of its decision boundaries in the vicinity of input and in all relevant subspaces.

Let's now take one step further and constrain his education to remain 11th grade and re-examine the effect of race. The resulting closest flip points are shown in Table 3 for two cases: where his race is kept Black and where it is changed to Asian. Clearly, being Asian requires considerably smaller changes in other features in order to reach the decision boundary of the model and flip to the high income class. This shows that race can be an influential feature in the model's classifications of people with low education. Having education above the 12th grade for this individual makes the effect of race negligible.

Table 3: Race can be an influential feature for individuals with low education. Closest flip points for testing point #1 in Adult dataset when education is fixed to 7th grade and race is changed from Black to Asian.

Data	Closest flip point (Black)	Closest flip point (Asian)
Age	41.9	32.4
Years of education	7 (11th grade)	7 (11th grade)
Marital status	Married-ArmedForcesSpouse	Married-ArmedForcesSpouse
Relationship	Husband	Husband
Occupation	Tech-support	Tech-support
Hours-per-week of work	44.3	42.4

We further observe that gender does not have an effect on the model's classification for this individual, whether the education is high or low. The effect of other features related to occupation and family can also be studied.

5.3.2 Group-level auditing using flip points

As an example, we consider the group of people with native country of Mexico. About 95% of this population have income "$\leq$ 50K" and 77% of them are Male. We compute the closest flip points for this population and investigate the patterns in them and how frequently features have changed from data points to flip points, and in what way.

Let's consider the effect of gender. 99% of the females in this group have income "$\leq$ 50K" and for 40% of them, their closest flip point is Male. Among the Males, however, less than 1% have a Female flip point; some of these are high-income individuals for whom the change in gender flips them to low-income.

Let's now consider the patterns in flip points that change low income males and females to high-income. For occupation, the most common change is entering into the Tech-sector and the most common exit is from the Farming-fishing occupation. For relationship, the most common change is to being married and the most common exit is from being Not-in-family and Never-married. Among the continuous features, Years of education and Capital-gain have changed most frequently.

5.3.3 Group-level auditing using flip directions

Consider the subset of directions that flip a "$\leq$ 50K" income to "$>$ 50K" for the population with native country of Mexico. The first principal component reveals that, for this model and this population, the most prominent features with positive impact are having a higher education, having Capital-gain, and working in the Tech-sector, while the features with most negative impact are being Never-married, being Female, and having Capital-loss. Looking more deeply at the data, pivoted QR decomposition of the matrix of flip directions reveals that some features, such as being Black and native country of Peru have no impact on this flip.

5.3.4 Group-level analysis of flip directions for misclassifications

Besides studying specific groups of individuals, we can also study the misclassifications of the model. PCA on the flip directions for all the misclassified points in the training set shows that Age has the largest coefficient in the first principal component, followed by Hours-per-week of work. The most significant feature with negative coefficient is having Capital-gain. These features can be considered the most influential in confusing and de-confusing the model. PCA on the flip directions explains how our model is influenced by various features and its vulnerabilities for misclassification. It thus enables us to create

inputs that are mistakenly classified for adversarial purposes, as explained by Lakkaraju and Bastani [19] and Slack et al. [39].

6 Comparison with other interpretation approaches for deep learning

Our use of flip points for interpretation and debugging builds on existing methods in the literature but provides more comprehensive capabilities. For example, Spangher et al. [40] compute flip sets only for linear classifiers and do not use them to explain the overall behavior of the model, identify influential features, or debug.

LIME [31] and Anchors [32] rely on sampling around an input in order to investigate decision boundaries, inefficient and less accurate than our approach, and the authors do not propose using their results as we do. LIME provides a coefficient for each feature (representing a hyperplane) which may not be easily understandable by non-experts (e.g., a loan applicant or a clinician), especially when dealing with a combination of discrete and continuous features. LIME's approach also relies on simplifying assumptions, such as the ability to approximate decision boundaries by hyperplanes, which leads to contradictions between the LIME output and the model output [46], a.k.a. infidelity. So, our method has an accuracy advantage over their method, too. Moreover, their reliance on random perturbations of data points can be considered a computational limitation when applying their method to deep learning models.

The interpretation we provide for nonlinear deep learning models is comparable in quality and extent to the interpretations provided in the literature for simple models. For example, the model suggested by [3] for the FICO Explainable ML dataset reports the most influential features in decision making of their model, similar to our findings in Section 5.1, and investigates the overall behavior of the model, similar to our results for the Adult dataset. But, their methods are not applicable for auditing deep learning models. Moreover, they do not provide a detailed explanation report.

We also show how decision boundaries can be altered to change the behavior of models, an approach not previously explored for deep learning models.

7 Conclusions and future work

We proposed the computation of flip points and developed methods to thoroughly audit and debug deep learning models with continuous output. We demonstrated that computation of the closest flip point to an input provides useful information to the user, explaining why a model produced a particular output and identifying any small changes in the input that would change the output. Flip points also provide useful information to model auditors, exposing bias and revealing patterns in misclassifications. We provided an algorithm to

formalize the auditing procedure for individuals and also for groups. Finally, we showed how model developers can use flip points in order to alter the decision boundaries and eliminate undesirable behavior of models.

This is the first work to use advanced numerical optimization techniques and state-of-the-art methods in numerical linear algebra, such as rank determination and reduced-order models, to analyze the behavior and biases in deep learning models and to improve them. Our examples involved neural networks, but our ideas are applicable to any black-box classification model. We would still compute flip points by minimizing (8), subject to (3)-(6), but our choice of optimization method might change.

Our proposed method has accuracy advantages over existing methods in the literature (such as LIME). We also showed why the common counterfactual formulation in the literature is mathematically ill-defined and does not yield the closest point on the decision boundary of a model. We introduced the concept of constrained flip points in order to compute closest boundary point in certain sub-spaces, and showed how such points can provide insights about the model behavior and communicate with non-expert users (such as loan applicant or a clinician) via an explanation report. By examining constrained flip points in various subspaces of domain, we showed that certain behaviors and biases of a model towards features such as race, may be hidden in certain subspaces, not easily recognizable by common methods in the literature.

For future work, we would consider models with continuous outputs other than classification models, for example, a model that recommends the dose of a drug for patients. Other directions of research include auditing text analysis or drug dosage models. Our methods can promote accountability and transparency in deep learning models.

8 Policy implications

Our work has policy implications for auditing machine learning models. While there has been many calls for legislation and regulations to require accountability and transparency about the use of machine learning models [29], sometimes the same publications that advocate for such regulations use improper mathematical procedures for auditing the models.

From the public policy perspective, the mere requirement for auditing the models is not sufficient; rather it is equally important that the auditing method itself is sound and accurate. To ensure that, we need to rely on sound mathematical principles and algorithms. Therefore, applied mathematicians should be main contributors in defining the auditing procedures and setting the standards.

This is evident in regulations imposed in more established fields. For example, for designing a bridge structure, federal and local agencies have requirements on which computational procedures should be followed to analyze and design the bridge. There are detailed guidelines grounded in engineering mechanics and applied mathematics. For example, using the finite element

method to analyze and design a bridge structure is not a choice, but a requirement [1]. A bridge structure analyzed and designed by simplified methods such as linear regression will not receive approval, even if the bridge designer believes linear regression is a good enough approach and finite element analysis is an overkill.

But that is not the current state of affairs in machine learning. Currently, any auditor may feel free to use any auditing method that they desire. One might make the assumption that the decision boundaries are approximable by hyper-planes, fit a linear regression model, and use the coefficients of the hyper-plane to explain which features are more influential in model's decisions. Another auditor might decide to use an ill-defined optimization problem to explain the behavior of the model. Some other auditor might decide to use first-order Taylor approximation to audit their model. Other auditor may create a simple model to partially emulate the original model and then audit the simple model instead. Evidence has suggested again and again that those simplifications are usually unjustified, and in many cases they are not even computationally beneficial: e.g., solving an ill-defined optimization problem is not faster nor less expensive than solving its well-defined formulation, and solving an optimization problem with random perturbations is more challenging and its solution is less reliable than solving it with proper optimization algorithms.

In this paper, we set the basic and proper standard for auditing these models. We explain that a classification model is a function that partitions its domain and assigns a class to each partition. Partitions are defined by its decision boundaries and so is the model. We formulate the proper optimization problems to compute exact points on those boundaries. We explain the challenges that one might encounter when solving such optimization problems and also explain how one may overcome those issues. We then explain the necessity of investigating the decision boundaries in various sub-spaces in the domain. We extend those procedures to group-auditing. These procedures should be an inescapable part of auditing any black-box classification model.

References

1. American Association of State Highway and Transportation Officials (2020) AASHTO LRFD Bridge Design Specifications, 9th edn
2. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2):157–166
3. Chen C, Lin K, Rudin C, Shaposhnik Y, Wang S, Wang T (2018) An interpretable model with globally consistent explanations for credit risk. arXiv preprint arXiv:181112615
4. Dong J, Rudin C (2020) Exploring the cloud of variable importance for the set of all good models. Nature Machine Intelligence 2(12):810–824

5. Dua D, Graff C (2017) UCI machine learning repository. URL http://archive.ics.uci.edu/ml
6. Dunlavy DM, O'Leary DP (2005) Homotopy optimization methods for global optimization. Tech. rep., Sandia National Laboratories
7. Elsayed G, Krishnan D, Mobahi H, Regan K, Bengio S (2018) Large margin deep networks for classification. In: Advances in Neural Information Processing Systems, pp 842–852
8. Fawzi A, Moosavi-Dezfooli SM, Frossard P (2017) The robustness of deep networks: A geometrical perspective. IEEE Signal Processing Magazine 34(6):50–62
9. Fawzi A, Moosavi-Dezfooli SM, Frossard P, Soatto S (2018) Empirical study of the topology and geometry of deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3762–3770
10. Golub GH, Van Loan CF (2012) Matrix Computations, 4th edn. JHU Press, Baltimore
11. Hanin B (2018) Which neural net architectures give rise to exploding and vanishing gradients? In: Advances in Neural Information Processing Systems (NeurIPS 2018), pp 580–589
12. Ilyas A, Santurkar S, Engstrom L, Tran B, Madry A (2019) Adversarial examples are not bugs, they are features. Advances in Neural Information Processing Systems 32
13. Jetley S, Lord N, Torr P (2018) With friends like these, who needs adversaries? In: Advances in Neural Information Processing Systems, pp 10749–10759
14. Jiang Y, Krishnan D, Mobahi H, Bengio S (2019) Predicting the generalization gap in deep networks with margin distributions. In: International Conference on Learning Representations
15. Johnson SG (2014) The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt
16. Koh PW, Liang P (2017) Understanding black-box predictions via influence functions. In: International Conference on Machine Learning, pp 1885–1894
17. Koh PW, Ang KS, Teo HH, Liang P (2019) On the accuracy of influence functions for measuring group effects. In: Advances in Neural Information Processing Systems, pp 5254–5264
18. Lage I, Chen E, He J, Narayanan M, Kim B, Gershman S, Doshi-Velez F (2019) An evaluation of the human-interpretability of explanation. arXiv preprint arXiv:190200006
19. Lakkaraju H, Bastani O (2020) "How do I fool you?": Manipulating user trust via misleading black box explanations. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, pp 79–85
20. Lakkaraju H, Kamar E, Caruana R, Leskovec J (2017) Interpretable & explorable approximations of black box models. arXiv preprint arXiv:170701154

21. Lakkaraju H, Kamar E, Caruana R, Leskovec J (2019) Faithful and customizable explanations of black box models. In: Artificial Intelligence, Ethics, and Society
22. Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp 4765–4774
23. Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, Katz R, Himmelfarb J, Bansal N, Lee SI (2020) From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence 2(1):2522–5839
24. Mobahi H (2016) Training recurrent neural networks by diffusion. ArXiv e-prints abs/1601.04114
25. Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2574–2582
26. Mothilal RK, Sharma A, Tan C (2020) Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp 607–617
27. Narayanan M, Chen E, He J, Kim B, Gershman S, Doshi-Velez F (2018) How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. arXiv preprint arXiv:180200682
28. Nocedal J, Wright S (2006) Numerical Optimization, 2nd edn. Springer, New York
29. O'Neil C (2016) Weapons of math destruction: How big data increases inequality and threatens democracy. Crown Publishers, New York
30. Ramachandran P, Zoph B, Le QV (2018) Searching for activation functions. arXiv preprint arXiv:171005941
31. Ribeiro MT, Singh S, Guestrin C (2016) Why should I trust you?: Explaining the predictions of any classifier. In: International Conference on Knowledge Discovery and Data Mining, ACM, pp 1135–1144
32. Ribeiro MT, Singh S, Guestrin C (2018) Anchors: High-precision model-agnostic explanations. In: AAAI Conference on Artificial Intelligence, pp 1527–1535
33. Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 1(5):206–215
34. Rudin C, Ertekin Ş (2018) Learning customized and optimized lists of rules with mathematical programming. Mathematical Programming Computation 10(4):659–702
35. Rudin C, Radin J (2019) Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition. Harvard Data Science Review 1(2)
36. Rudin C, Passonneau RJ, Radeva A, Dutta H, Ierome S, Isaac D (2010) A process for predicting manhole events in Manhattan. Machine Learning 80(1):1–31

37. Rudin C, Wang C, Coker B (2020) The age of secrecy and unfairness in recidivism prediction. Harvard Data Science Review 2(1)
38. Russell C (2019) Efficient search for diverse coherent explanations. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp 20–28
39. Slack D, Hilgard S, Jia E, Singh S, Lakkaraju H (2019) How can we fool LIME and SHAP? Adversarial attacks on post hoc explanation methods. arXiv preprint arXiv:191102508
40. Spangher A, Ustun B, Liu Y (2018) Actionable recourse in linear classification. In: Proceedings of the 5th Workshop on Fairness, Accountability and Transparency in Machine Learning
41. Tsipras D, Santurkar S, Engstrom L, Turner A, Madry A (2019) Robustness may be at odds with accuracy. In: International Conference on Learning Representations
42. Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106(1):25–57
43. Wachter S, Mittelstadt B (2019) A right to reasonable inferences: Rethinking data protection law in the age of big data and AI. Columbia Business Law Review (2):494–620
44. Wachter S, Mittelstadt B, Russell C (2018) Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harvard Journal of Law & Technology 31(2)
45. Wexler J, Pushkarna M, Bolukbasi T, Wattenberg M, Viégas F, Wilson J (2019) The what-if tool: Interactive probing of machine learning models. IEEE Transactions on Visualization and Computer Graphics 26(1):56–65
46. White A, Garcez Ad (2019) Measurable counterfactual local explanations for any classifier. arXiv preprint arXiv:190803020
47. Yeh IC, Lien Ch (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):2473–2480
48. Yousefzadeh R (2019) Interpreting machine learning models and application of homotopy methods. PhD thesis, University of Maryland, College Park
49. Yousefzadeh R, O'Leary DP (2019) Investigating decision boundaries of trained neural networks. arXiv preprint arXiv:190802802
50. Yousefzadeh R, O'Leary DP (2020) Deep learning interpretation: Flip points and homotopy methods. In: Proceedings of Machine Learning Research, vol 107, pp 1–26
51. Zeng J, Ustun B, Rudin C (2017) Interpretable classification models for recidivism prediction. Journal of the Royal Statistical Society: Series A (Statistics in Society) 180(3):689–722

A Description of variables for the FICO dataset

The name of each variable for the FICO dataset can be viewed in the first column of Table A1. The second column shows the corresponding description for each variable as provided by FICO. Additionally, the third column of this table shows the value of each variable for data point #1. Detailed information about the challenge can be found here: `https://community.fico.com/s/explainable-machine-learning-challenge`.

B Code

The code along with a readme file, an example procedure, and our homotopy algorithm are available at `https://github.com/roozbeh-yz/auditing`.

Besides our homotopy algorithm, the NLOPT package has more than 30 optimization algorithms, as explained here: `https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms`.

For the Matlab optimization toolbox, we used its fmincon function, and the list of its optimization algorithms is available here: `https://www.mathworks.com/help/optim/ug/choosing-the-algorithm.html#bsbwxm7`.

IPOPT uses an interior point algorithm as explained here: `https://coin-or.github.io/Ipopt`.

C Information about the models

Here, we provide more information about the models we have trained and used in Section 5. We have used fully connected feed-forward neural networks with up to 6 hidden layers. The number of neurons for the models used for each data set is shown in Table C1. Note that the input layers do not have any neurons, instead they have simple nodes representing the features in the data domain. The activation function we have used for the neurons is the error function, as defined in [50]. We have also used softmax on the output layer, and cross entropy for the loss function.

Datasets we have used are standard and commonly used as benchmarks in the machine learning literature. The testing accuracy of models we have trained are all either equivalent or better than the testing accuracy of best models in the literature. For example, Spangher et al. [40] reports testing accuracy of 80.6% for the model trained on the Credit dataset, while we achieve 81.8% testing accuracy on the same dataset with the same cross-validation. For the FICO dataset, Chen et al. [3] which was the winner of the dataset challenge reported testing accuracy of 69.96% for their neural network model, while testing accuracy of our model is 72.9%. The testing accuracy of our

model on Adult Income dataset is equivalent to the best accuracy reported in the literature.[9]

While erf is not a very common choice for activation function, it has been shown that its performance in terms of accuracy can be comparable to other activation functions [30]. Mobahi [24] has also reported success in using the erf for training recurrent neural networks.

Our auditing and debugging methods are applicable to deep learning models, regardless of the model architecture, activation function, and the method used for training. A trained model is a classification function defined by its decision boundaries. Using our methods, one can gain insights about those decision boundaries and alter them.

When an auditor is in the position to audit a model in the real world, they might only have access to inputs and outputs, and the gradients of the model, which are basically what we need to solve our optimization problem and proceed with auditing and debugging. Having limited access to the model and limited knowledge about its background can be common in practice, as considered by Rudin et al. [37]. Moreover, as noted by Dong and Rudin [4], auditing a model and making statements about its behavior is different from making statements about the data itself. Here, we developed computational methods for the former.

[9] Some papers have reported slightly better accuracy on the Adult dataset, but those are on a reduced dataset.

Table A1: Variable descriptions for the FICO dataset.

Variable name	Description	data point #1
ExternalRiskEstimate	Consolidated version of risk markers	55
MSinceOldestTradeOpen	Months Since Oldest Trade Open	144
MSinceMostRecentTradeOpen	Months Since Most Recent Trade Open	4
AverageMInFile	Average Months in File	84
NumSatisfactoryTrades	Number of Satisfactory Trades	20
NumTrades60Ever2DerogPubRec	Number of Trades 60+ Ever	3
NumTrades90Ever2DerogPubRec	Number of Trades 90+ Ever	0
PercentTradesNeverDelq	Percentage of Trades Never Delinquent	83
MSinceMostRecentDelq	Months Since Most Recent Delinquency	2
MaxDelq2PublicRecLast12M	Max Delinquency/Public Records in the Last 12 Months	3
MaxDelqEver	Max Delinquency Ever	5
NumTotalTrades	Number of Total Trades (total number of credit accounts)	23
NumTradesOpeninLast12M	Number of Trades Open in the Last 12 Months	1
PercentInstallTrades	Percentage of Installment Trades	43
MSinceMostRecentInqexcl7days	Months Since Most Recent Inquiry (excluding last 7 days)	0
NumInqLast6M	Number of Inquiries in the Last 6 Months	0
NumInqLast6Mexcl7days	Number of Inquiries in the Last 6 Months (excluding last 7 days)	0
NetFractionRevolvingBurden	Net Fraction Revolving Burden	33
NetFractionInstallBurden	Net Fraction Installment Burden	-8
NumRevolvingTradesWBalance	Number of Revolving Trades with Balance	8
NumInstallTradesWBalance	Number of Installment Trades with Balance	1
NumBank2NatlTradesWHighUtil	Number of Bank/National Trades with High Utilization Ratio	1
PercentTradesWBalance	Percentage of Trades with Balance	69

Table C1: Number of neurons in neural network used for each data set.

Data set	FICO	Credit	Adult
Input layer	20	28	88
Layer 1	13	14	40
Layer 2	9	9	32
Layer 3	6	8	24
Layer 4	5	8	20
Layer 5	4	7	16
Layer 6	-	-	14
Output layer	2	2	2