Set-Valued Sensitivity Analysis of Deep Neural Networks

Xin Wang[1], **Feilong Wang**[2], **Xuegang (Jeff) Ban**[1]

[1]Department of Civil and Environmental Engineering, University of Washington, Seattle, WA, 98195, United States
[2]School of Transportation and Logistics, Southwest Jiaotong University, Chengdu, 610032, China
xinw22@uw.edu, Flwang@swjtu.edu.cn, banx@uw.edu

Abstract

This paper proposes a sensitivity analysis framework based on set-valued mapping for deep neural networks (DNN) to understand and compute how the solutions (model weights) of DNN respond to perturbations in the training data. As a DNN may not exhibit a unique solution (minima) and the algorithm of solving a DNN may lead to different solutions with minor perturbations to input data, we focus on the sensitivity of the solution set of DNN, instead of studying a single solution. In particular, we are interested in the expansion and contraction of the solution set in response to data perturbations. If the change of solution set can be bounded by the extent of the data perturbation, the model is said to exhibit the Lipschitz-like property. This 'set-to-set' analysis approach provides a deeper understanding of the robustness and reliability of DNNs during training. Our framework incorporates both isolated and non-isolated minima, and critically, does not require the Hessian of loss function being non-singular. By developing set-level metrics such as distance between sets, convergence of sets, derivatives of set-valued mapping, and stability across the solution set, we prove that the solution set of the Fully Connected Neural Network holds Lipschitz-like properties. For general neural networks (e.g. Resnet), we also develop a novel method to estimate the new solution set following data perturbation without retraining.

Extended version — https://arxiv.org/abs/2412.11057

Introduction

Sensitivity analysis is a classic topic that crosses optimization (Fiacco 1983), machine learning, and deep learning (Yeung et al. 2010; Koh and Liang 2017; Christmann and Steinwart 2004). It studies how the solution of a *model* responds to minor perturbations in hyperparameters or input data. For example, Christmann and Steinwart (2004) proved that the solution of a classification model with convex risk function is robust to bias in the data distribution. Here we focus on deep neural networks (DNN), for which the solution is the weights of DNN trained (optimized) on input data. We are concerned about how the solution of DNN responds to perturbations in the input data in the *training* stage of the model.

In the domain of deep learning (e.g., DNN), sensitivity analysis has drawn attention due to its wide range of applications, such as designing effective data poisoning attacks (Muñoz-González et al. 2017; Mei and Zhu 2015), evaluating the robustness of models (Weng et al. 2018), and understanding the impact of important features in training data on the prediction (Koh and Liang 2017). Define a neural network $f : \mathcal{X} \rightrightarrows \mathcal{Y}$, where $\mathcal{X}$ (e.g., images) is the input space and $\mathcal{Y}$ (e.g., labels) is the output space. Given training data samples $x = [x_1, x_2, \ldots, x_n]$ and $y = [y_1, y_2, \ldots, y_n]$, and the loss function L, the empirical risk minimizer is given by $\hat{w} \overset{\text{def}}{=} \arg\min_{w\in\mathcal{W}} \frac{1}{n}\sum_{i=1}^{n} L\left(x_i, y_i, w\right)$. This paper assumes that we perturb only the features, keeping the label constant. The learning process from data to local minimizers thus can be formulated as a set-valued mapping $S : \mathcal{X}^n \rightrightarrows \mathcal{W}$ [1],

$$S(x) = \left\{ \hat{w} | \hat{w} \overset{\text{def}}{=} \arg\min_{w\in\mathcal{W}} \frac{1}{n}\sum_{i=1}^{n} L\left(x_i, y_i, w\right) \right\}. \quad (1)$$

For the unperturbed dataset $\bar{x} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n]$ and $\bar{w} \in S(\bar{x})$, current sensitivity analysis (Koh and Liang 2017; Nickl et al. 2024; Christmann and Steinwart 2004) aims to study the change of $S(\bar{x})$ when an individual point $\bar{x}_p$ is perturbed to x_p. Specifically, if the data $\bar{x}$ is perturbed to $\hat{x} = [\bar{x}_1, \bar{x}_2, \ldots, x_p, \ldots, \bar{x}_n]$, the sensitivity analysis can be conducted by examining a limit:

$$\lim_{x_p \to \bar{x}_p} \frac{S\left(\hat{x}\right) - S(\bar{x})}{\|x_p - \bar{x}_p\|}. \quad (2)$$

Most current sensitivity analysis methods for DNN suffer from the following two issues. First, to figure out the sensitivity of solution w w.r.t. data x_p, one of the most common approaches (e.g., the influence function (Koh and Liang 2017)) is to apply the Dini implicit function theorem (Dontchev and Rockafellar 2009) to the optimality condition of the model: $\frac{1}{n}\sum_{i=1}^{n} \nabla_w L\left(\bar{x}_i, \bar{y}_i, \bar{w}\right) = 0$, which leads to:

$$\begin{aligned} \lim_{x_p \to \bar{x}_p} \frac{S\left(\hat{x}\right) - S(\bar{x})}{\|x_p - \bar{x}_p\|} &= \nabla_{x_p} w \Big|_{x_p = \bar{x}_p} \\ &= -H_{\bar{w}}^{-1} \nabla_{x_p} \nabla_\theta L(\bar{x}_p, \bar{y}_p, \bar{w}), \end{aligned} \quad (3)$$

where $H_{\bar{w}}^{-1} = \frac{1}{n}\sum_{i=1}^{n} \nabla_w^2 L\left(\bar{x}_i, \bar{y}_i, \bar{w}\right)$ is the Hessian. However, the application of the Dini implicit function theorem cannot apply when the Hessian $H_{\bar{w}}$ is not invertible

[1]Here, $\mathcal{X}^n$ denotes the n-fold Cartesian product of $\mathcal{X}$, i.e., $\mathcal{X}^n = \mathcal{X} \times \mathcal{X} \times \cdots \times \mathcal{X}$ (n times).

due to the often non-locally strong convex loss function L of DNN (Li et al. 2018). Second, the current approaches (Koh and Liang 2017; Nickl et al. 2024) assume S is a single-valued mapping, omitting the fact that S is often set-valued. Research has shown that DNN may not exhibit a unique solution $S(x)$ even when L is convex. For example, it was noticed in Li et al. (2018) that the stochastic gradient descent (SGD) method can find flat minima (solutions located in the flat valley of the loss landscape); others found that all SGD solutions for DNN may form a manifold (Benton et al. 2021; Cooper 2021). When an application (e.g., a data poisoning attack) is designed and evaluated based on only one of the solutions, it overlooks the fact that the learning algorithm may converge to other solutions during re-training.

In this paper, we incorporate the fact that S is often set-valued into the sensitivity analysis framework for DNNs. This extends the scope of sensitivity analysis in DNNs from focusing on a single solution to a solution set, shifting from the traditional 'point-to-point' analysis to a 'set-to-set' paradigm. That is, we study how the solution set of a DNN expands and contracts in response to data perturbations. The proposed approach covers more general situations in risk minimization of DNN, including isolated local minima, non-isolated minima, and minima that consist of a connected manifold. More importantly, it directly deals with the solution sets without the assumption of non-singular Hessian matrix, offering a more complete understanding of DNN.

Considering S as set-valued, with the isolated minima a special case where solution set $S(x)$ contains only one element, the sensitivity analysis aims to study i) whether the limit (2) exists, and ii) whether the limit (2) is bounded. However, directly studying the limit when mapping S is set-valued can be challenging. On the one hand, $S(x)$ may be a large or even infinitely large set (e.g., a manifold), which makes it complicated to analyze. To address this, we instead focus on the change of the solution set $S(x)$ within a small neighborhood. Given a pair $(\bar{x}, \bar{w}), \bar{w} \in S(\bar{x})$, we study the change of $S(x) \cap U$, where U is a neighborhood of $\bar{w}$, in response to the data perturbation within a neighborhood V of $\bar{x}$. On the other hand, as $S(x)$ and $S(\bar{x})$ are sets rather than single solutions, the subtraction operation between sets does not exist (set operations only include union, intersection, and set difference), which invalidates the limit (2). Consequently, we leverage the *Lipchitz-like* property of S around $(\bar{x}, \bar{w})$, to measure the change of $S(x) \cap U$. We say that S holds *Lipchitz-like* property around $(\bar{x}, \bar{w}) \in \operatorname{gph} S := \{(x, w) \mid w \in S(x)\}$, if there exist neighborhoods U of $\bar{w}, V$ of $\bar{x}$ and a positive real number κ such that (Dontchev and Rockafellar 2009) [2]

$$S(x') \cap U \subset S(x) + \kappa \|x - x'\| \mathbb{B}, \quad \forall x', x \in V, \qquad (4)$$

where $\mathbb{B}$ is a closed unit ball. The Lipschitz-like property is an extension of Lipschitz continuity defined for single-valued functions to set-valued mappings. κ, as a scalar, is the *Lipschitz modulus* that describes the upper bound of the solution set's change in response to data perturbations. The existence of the limit (2) can be interpreted as establishing a Lipschitz-like property. The bound of the limit (2) is equivalent to a bounded κ that characterizes how sensitive the model solution is w.r.t input data, defined for the training stage. It complements the studies on Lipschitz constants of DNNs (Fazlyab et al. 2019; Virmaux and Scaman 2018) that have been only defined and studied so far for the inference stage of DNNs (with fixed model solutions).

In our "set-to-set" analysis framework, the study of existence and boundedness of limits (2) is transferred as the following question: Given a local minimizer $\bar{w}$ with its neighborhoods U, and training data $\bar{x}$ with its neighborhoods V, where $(\bar{x}, \bar{w}) \in \operatorname{gph} S$, does the solution mapping S satisfy the Lipschitz-like property in a neighborhood of $(\bar{x}, \bar{w})$, and if so, how can we estimate the associated bounded Lipschitz modulus? Moreover, we also explore how, when we perturb the data $\bar{x}$ to x^p, with $x^p \in V$, we can identify the new solution set $S(x^p) \cap U$ without re-training.

We prove that, for the Deep Fully Connected Neural Network (DFCNN) with the Relu activation function, the solution set S holds the Lipschitz-like property. A bound of the Lipschitz modulus is also provided. This reveals that the solution set of a DFCNN will not deviate significantly when there are biases in the training data, allowing us to estimate the new solution set based on the behavior of S around $(\bar{x}, \bar{w})$. We also introduce *graphical derivatives* to capture the local linear approximation of S around $(\bar{x}, \bar{w})$ for a DNN (not only for DFCNN). The graphical derivative based method can accurately estimate $S(x^p) - \bar{w}$, providing solutions following data perturbations for a DNN (e.g., Resnet56) with near zero training loss. In particular, when $S(x^p)$ is single-valued, i.e. the solution set $S(x^p)$ within neighborhood U includes only one solution, our graphical derivative based method is equivalent to the influence function (Koh and Liang 2017).

The contribution of our paper is summarized as follows:

1) We introduce a set-valued mapping approach to understand the sensitivity of solutions of DNN in relation to perturbations in the training data. Our framework accommodates both isolated and non-isolated minima without relying on convex loss assumption.

2) We prove that the solution mapping of DFCNN holds the Lipschitz-like property (and thus stable during training) and estimate a bound for the Lipschitz modulus.

3) We propose a graphical-derivative-based method to estimate the new solution set of a DNN when the training data are perturbed, and simulate it using the Resnet56 with the CIFAR-10 dataset.

Preliminary Knowledge

This section briefly summarizes the necessary preliminary knowledge for sensitive analysis of set-valued mapping, covering the distance between sets, sets convergence, and the generalized derivative. These concepts will be used to characterize the behavior of set-valued mapping and play a key role in defining the criteria for Lipschitz-like property.

Definition 1. (Distance between sets) *The distance from a point w to a set C is*

$$d_C(w) = d(w, C) = \inf_{w' \in C} |w - w'|.$$

[2] Here, $\|x - x'\|$ represents the 2-norm of $(x - x')$.

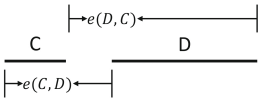


Figure 1: Illustration of $e(C, D)$ and $e(D, C)$, where C and D are two closed sets. The Pompeiu-Hausdorff Distance of C beyond D is $h(C, D) = \max\{e(C, D), e(D, C)\} = e(D, C)$ for this example.

For sets C and D, the Pompeiu-Hausdorff Distance of C beyond D is defined by

$$h(C, D) = \max\{e(C, D), e(D, C)\},$$

where

$$e(C, D) = \sup_{w \in C} d(w, D), e(D, C) = \sup_{w \in D} d(w, C).$$

An illustrative figure is provided in Figure 1, where C and D are two segments.

Definition 2. (Painlevé-Kuratowski Set convergence) *Given a set-valued mapping $S : X \rightrightarrows W$, the Painlevé-Kuratowski outer set limit as $x \to \bar{x}$ is*

$$\limsup_{x \to \bar{x}} S(x) := \{w \in W \mid \exists \textit{ sequences } x_k \to \bar{x} \textit{ such that } w_k \in S(x_k) \to w\}.$$

the Painlevé-Kuratowski inner set limit as $x \to \bar{x}$ is

$$\liminf_{x \to \bar{x}} S(x) := \{w \in W \mid \forall \textit{ sequences } x_k \to \bar{x}, w_k \in S(x_k) \to w\}. \quad (5)$$

Definition 3. *A vector η is tangent to a set Γ at a point $\bar{\gamma} \in \Gamma$, written $\eta \in T_\Gamma(\bar{\gamma})$, if*

$$\frac{\gamma_i - \bar{\gamma}}{\tau_i} \to \eta \textit{ for some } \gamma_i \to \bar{\gamma}, \gamma_i \in \Gamma, \tau_i \searrow 0.$$

Where $T_\Gamma(\bar{\gamma})$ is the tangent cone to Γ at $\bar{\gamma}$.

Definition 4. *Given a convex set Γ in $\mathbb{R}^n$ and a point $\bar{\gamma}$ in Γ, the normal cone to Γ at $\bar{\gamma}$, denoted $N_\Gamma(\bar{\gamma})$, is defined as the set of all vectors $\xi \in \mathbb{R}^n$ that satisfy the condition:*

$$N_\Gamma(\bar{\gamma}) = \{\xi \in \mathbb{R}^n : \langle \xi, \gamma - \bar{\gamma} \rangle \leq 0 \textit{ for all } \gamma \in \Gamma\}.$$

Remark 1: This study only focuses on the normal cone and tangent cone of convex sets. Refer to Appendix C for an illustrative diagram of Definition 3 and Definition 4.

We next define the generalized derivative that includes both the graphical derivative and coderivative. The graphical derivative is a concept used in the analysis of set-valued mappings (multifunctions). It generalizes the derivative of functions to set-valued mappings, capturing the local behavior of a set-valued mapping around a particular point. The coderivative complements the graphical derivative by offering a reverse perspective: instead of describing how outputs respond to changes in inputs, it examines how inputs need to change to achieve specific output behaviors.

Definition 5. (Generalized derivatives)[3] *Consider a mapping $S : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ and a point $\bar{x} \in \operatorname{dom} S$. The **graphical derivative** of S at $\bar{x}$ for any $\bar{w} \in S(\bar{x})$ is the mapping $DS(\bar{x} \mid \bar{u}) : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ defined by*

$$v \in DS(\bar{x} \mid \bar{w})(\mu) \Longleftrightarrow (\mu, v) \in T_{\operatorname{gph} S}(\bar{x}, \bar{w}),$$

*whereas the **coderivative** is the mapping $D^*S(\bar{x} \mid \bar{u}) : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ defined by*

$$q \in D^*S(\bar{x} \mid \bar{w})(p) \Longleftrightarrow (q, -p) \in N_{\operatorname{gph} S}(\bar{x}, \bar{w}).$$

***Remark 1:** Graphical derivative can also be expressed as:*

$$DS(\bar{x} \mid \bar{w})(\mu) = \limsup_{\substack{\tau \searrow 0 \\ \mu_k \to \mu}} \frac{S(\bar{x} + \tau\mu_k) - \bar{w}}{\tau} \quad (6)$$

***Remark 2:** In the case of a smooth, single-valued mapping $F : \mathbb{R}^n \to \mathbb{R}^m$, one has*

$$\begin{aligned} DF(\bar{x})(\mu) &= \nabla F(\bar{x})\mu \textit{ for all } \mu \in \mathbb{R}^n \\ D^*F(\bar{x})(p) &= \nabla F(\bar{x})^* p \textit{ for all } p \in \mathbb{R}^m \end{aligned} \quad (7)$$

Lipschitz-like Property of Deep Fully Connected Neural Network

This section studies the Lipschitz-like property of DNNs. We focus on DFCNN, a classical DNN, to demonstrate our main theorem results. For each data point $x_i \in x, x_i \in \mathbb{R}^d$, the first layer's weight matrix of a DFCNN is denoted as $W^{(1)} \in \mathbb{R}^{m \times d}$, and for each subsequent layer from 2 to H, the weight matrices are denoted as $W^{(h)} \in \mathbb{R}^{m \times m}$. $a \in \mathbb{R}^m$ is the output layer and the Relu function is given by $\sigma(\cdot)$. We recursively define a DFCNN, starting with ${x_i}^{(0)} = x_i$ for simplicity.

$$\begin{aligned} {x_i}^{(h)} &= \sigma\left(W^{(h)} {x_i}^{(h-1)}\right), 1 \leq h \leq H \\ f(x_i, w) &= a^\top {x_i}^{(H)}. \end{aligned} \quad (8)$$

Here ${x_i}^{(h)}$ is the output of the h-th layer. We denote $W := (W^{(1)}, \ldots, W^{(H)})$ as the weights of the network and $w := (w^{(1)}, \ldots, w^{(H)})$ as the vector of the flatten weights. In particular, $w^{(h)}$ is the vector of the flattened h-th weight $W^{(h)}$. Denote $dim(w^{(h)}) = p^{(h)}$ and $dim(w) = \sum_{i=1}^{H} p^{(h)} = p$.

For a DFCNN, we develop our method using the quadratic loss function: $L(x_i, y_i, w) = \frac{1}{2}\left(f\left(w, x_i\right) - y_i\right)^2$. w, as the neural network weights, is a local/global minimum of empirical loss $\frac{1}{n}\sum_{i=1}^{n} L\left(x_i, y_i, w\right)$. Since first-order optimization algorithms, such as SGD, are widely utilized, we employ the first-order optimality condition to characterize these minima. Let $R(x, y, w) = \nabla_w \frac{1}{n}\sum_{i=1}^{n} L\left(x_i, y_i, w\right)$. Since the label vector $y = [y_1, \ldots, y_n]$ is constant, we simplify the notation of $R(x, y, w)$ to $R(x, w)$. Then the solution of a DFCNN can be characterized by the set-valued mapping F:

$$F(x) = \{w | R(x, w) = \nabla_w \frac{1}{n}\sum_{i=1}^{n} L\left(x_i, y_i, w\right) = 0\}, \quad (9)$$

[3] A^* denotes the conjugate transpose of A, where A is a matrix

For layer h, we define mapping F_h as:

$$F_h(x) = \{w^{(h)} | R(x, w) = 0\}. \tag{10}$$

Following the classical sensitivity analysis (Koh and Liang 2017; Nickl et al. 2024), we first focus on one individual data $x_k \in x = [x_1, \ldots, x_n]$, which is perturbed. In this case, $F(x)$ and $F_h(x)$ in the above two equations are expressed as $F(x_k)$ and $F_h(x_k)$ to indicate that only x_k are perturbed. In the next section, we present the case with multiple data perturbations.

Assumption 1. *We assume that DNNs are overparameterized; under this assumption, a DFCNN has the capacity to memorize training data with zero training error, i.e. $p > d$.*

Assumption 2. *For given $\bar{x}$ and $\bar{w}$, $[\nabla_w R(\bar{x}, \bar{w}), \nabla_{x_k} R(\bar{x}, \bar{w})]$ is of full rank, where $\nabla_w R(\bar{x}, \bar{w}) \in \mathbb{R}^{p \times p}$, $\nabla_{x_k} R(\bar{x}, \bar{w}) \in \mathbb{R}^{p \times d}, p = dim(w), d = dim(x_k)$.*

Remark 1: A DNN with non-singular Hessian matrix, as a special case, satisfies Assumption 2. On the other hand, a matrix that satisifies Assumption 2 is not necessarily be non-signular.

Remark 2: In the following section, we will find that although some DNNs do not always satisfy Assumptions 1 or 2, our algorithm can still provide a reasonably accurate estimation of the sensitivity of the solution mapping.

The Lipschitz-like property and solution set estimation following data perturbation rely on the generalized derivative (see definition 5). Theorem 1 below provides an explicit formulation for the generalized derivative of F, enabling a convenient analysis of the local behavior of a solution mapping. It will be used in the proof of Theorem 2 that describes the Lipschitz-like property of the solution mapping.

Theorem 1. *For given $\bar{x}_k$ and $\bar{w}$, the graphical derivative and coderivative of F (defined by Definition 5) at $\bar{x}_k$ for $\bar{w}$ have the formulas:*

$$\begin{aligned} &DF\left(\bar{x}_k \mid \bar{w}\right)(\mu) = \{v \mid \nabla_w R(\bar{x}, \bar{w}) v + \nabla_{x_k} R(\bar{x}, \bar{w}) \mu = 0\} \\ &D^* F\left(\bar{x}_k \mid \bar{w}\right)(p) \\ &= \left\{q \mid (q, -p) = \left[\nabla_{x_k} R(\bar{x}, \bar{w}), \nabla_w R(\bar{x}, \bar{w})\right]^\top y\right\} \\ &\text{where} \quad y \in \mathbb{R}^{\dim(w)} \end{aligned} \tag{11}$$

Proof: see appendix A.

The following Theorem 2 then proves the Lipschitz-like property of DFCNN with the bound of the Lipshitz modulus. It reveals the potential training stability of DFCNN, i.e. the solution set will not change dramatically when perturbations are introduced to training data. The Lipschitz modulus is determined by the original solution $\bar{w}$ and input data $\bar{x}$.

Theorem 2. *For a given layer h ($1 \leq h \leq H$), mapping F_h holds the Lipschitz-like property. That is, for given $\bar{x}_k$ and $\bar{w}$, there exists neighborhoods V_k of $\bar{x}_k$ and U_h of $\bar{w}^{(h)}$, with a positive real number κ_h (the Lipschitz modulus) such that*

$$\begin{aligned} F_h\left(x_k'\right) \cap U_h \subset F_h(x_k) + \kappa_h \left\|x_k - x_k'\right\| \mathbb{B} \\ \forall x_k', x_k \in V_k \end{aligned} \tag{12}$$

where the the Lipschitz modulus κ_h can be expressed as:

$$\begin{aligned} \kappa_h = &\frac{\left\| \left[\prod_{k=1}^{H} W^{(k)^\top} \operatorname{diag}\left(\mathbf{1}\left(\sigma\left(W^{(k)} x_k^{k-1}\right) > 0\right)\right)\right] a \right\|}{\| \operatorname{diag}\left(\mathbf{1}\left(\sigma\left(W^{(h)} x_k^{h-1}\right) > 0\right)\right)} \\ &\left[\prod_{k=h+1}^{H} W^{(k)^\top} \operatorname{diag}\left(\mathbf{1}\left(\sigma\left(W^{(k)} x_k^{k-1}\right) > 0\right)\right)\right] \\ &a x_k^{(h-1)^\top} \|_F \end{aligned}$$

Proof: see appendix B.

The following theorem 3 generalizes theorem 2 by considering the shift of the entire dataset instead of perturbing a single point.

Theorem 3. *For given $\bar{x}$ and $\bar{w}$, there exists neighborhoods V of $\bar{x}$ and U of $\bar{w}^{(h)}$, with a positive real number κ (i.e., the Lipschitz modulus) such that*

$$F_h\left(x'\right) \cap U \subset F_h\left(x\right) + \kappa \left\|x - x'\right\| \mathbb{B} \quad \forall x', x \in V \tag{13}$$

Proof: see appendix B.

Remark: The Lipschitz modulus in Theorems 2 or 3 measures the extent of deviation of the solution set when the data points are perturbed within a neighborhood of $\bar{x}$. The Lipschitz modulus of DFCNN, which depends on the number of layers, the original solution, and the unperturbed data, can vary. However, it won't be unbounded even if it can be large, indicating that the solution set won't change dramatically.

Sensitivity Analysis for Solution Set

Theorem 3 reveals that the solution set of DFCNN after perturbation will not deviate dramatically from the original solution set, allowing us to approximate this change using the local information around $(\bar{x}, \bar{w})$. This section first proposes a method to estimate the new solution set of DFCNN, given the perturbation in the training data. Instead of only perturbing a single individual data point in the previous section, this section perturbs multiple data points simultaneously.

Consider a DFCNN with its weight $\bar{w}$, where $\bar{w}$ represents a solution obtained by a learning algorithm (e.g., SGD) trained on a set of pristine data $\bar{x} = \{\bar{x}_i\}, i \in I = \{1, 2, \ldots, n\}$. Assume the data is perturbed following $x_i^p = \bar{x}_i + \delta \Delta x_i$, where δ is the norm of perturbation, Δx_i is a unit vector that indicates the perturbation direction for point x_i. We denote by $\Delta x = \{\Delta x_i\}, i \in I$ the set of perturbations. To make the number of perturbed points more flexible, we set $K \subset I$ to denote the indices of the perturbed data, and let $\Delta x_i = 0$ for $i \in I \backslash K$. As defined by (1), $S(x^p)$ is the solution set of a DFCNN trained by the poisoned data $x^p = \bar{x} + \Delta x$ and $S(\bar{x})$ is the original solution set.

The graphical derivative $DS(\bar{x} \mid \bar{w})(\Delta x)$ captures how the solution w changes near $\bar{w}$ when x is perturbed in the direction of Δx. For any twice differentiable loss function $L(w)$ (such as the quadratic loss function), following theorem 1, $DS(\bar{x} \mid \bar{w})(\Delta x)$ is equivalent to (see Appendix A):

$$DS(\bar{x} \mid \bar{w})(\Delta x) = \{v \mid \nabla_w^2 \frac{1}{|I|} \sum_{i \in I} L(x_i, y_i, w)\, v + \frac{1}{|K|} \sum_{i \in K} \nabla_{x_i} \nabla_w L(x_i, y_i, w)\, \Delta x_i = 0\}. \tag{14}$$

The solution set $S(x^p)$ within U, a neighborhood of $\bar{w}$, can be estimated by:

$$S(x^p) \cap U \subset \bar{w} + DS(\bar{x} \mid \bar{w})(\Delta x) + o(\|\Delta x\|)\mathbb{B}. \tag{15}$$

In a special case, when the empirical risk function $\sum_{i=1}^{n} L(x_i, y_i, w)$ has a non-singular Hessian matrix, $DS(\bar{x} \mid \bar{w})(\Delta x)$ will include a unique v, indicating that the solution $\bar{w}$ can only move along direction v if we perturb $\bar{x}$ along direction Δx. This situation corresponds to the scenario where the solution $\bar{w}$ represents an isolated minimum. In this case, multiplying both sides of expression on the right side of (14) by the inverse of the Hessian matrix results in the influence function (Koh and Liang 2017).

Our set-based framework also provides a new evaluation method for the training stability of DNN, see Algorithm 1.

Algorithm 1: Training Stability Analysis Based on Set Valued Mapping

Input: Perturbation index set K, data perturbation Δx, original solution $\bar{w}$, unperturbed data $\bar{x}$, loss function L
Output: Distance between the new solution set and $\bar{w}$

1. Compute the graphical derivative $DS(\bar{x} \mid \bar{w})(\Delta x)$ following (14).
2. Estimate the new solution set by $S(x^p) \approx \bar{w} + DS(\bar{x} \mid \bar{w})(\Delta x)$
3. Calculate distance between the solution set and $\bar{w}$ following Pompeiu-Hausdorff Distance (see Definition 1)

Simulation for Solution Estimation

Although our theoretical results in the above two sections focus on DFCNN, this section demonstrates that the methods perform well for general DNNs. To show this, we next present two numerical examples to illustrate the proposed set-valued sensitivity analysis method. The first one is on a toy example and the second one is on the Resnet. All experiments are performed on an RTX 4090 GPU.

A Toy Example

We consider a linear neural network with 2 layers. Assume we only have two data points $(x_i, y_i), i = 1, 2$ and both x_i and y_i are real numbers. The solution set $S(x) = \{w = (w_1, w_2)\}$, where w_1, w_2 are the weights of the first and second layer, is given by minimizing the empirical risk :

$$(w_1, w_2) \triangleq \underset{w_1, w_2 \in R}{\operatorname{argmin}} \frac{1}{2}(y_1 - w_1 w_2 x_1)^2 + \frac{1}{2}(y_2 - w_1 w_2 x_2)^2. \tag{16}$$

Given the pristine dataset $(\bar{x}_1, \bar{y}_1) = (1, 2), (\bar{x}_2, \bar{y}_2) = (2, 4)$, the model solution constitutes a set as $w_1 * w_2 = 2$, and $\bar{w} = (1, 2)$ is obviously one of the solutions. We assume that the original solution converges to $\bar{w} = (1, 2)$ during training using this pristine data. We introduce perturbations to the data following the rule $x^p = \bar{x} + 0.2 * (-1, -2)$, and re-train the model using the poisoned data.

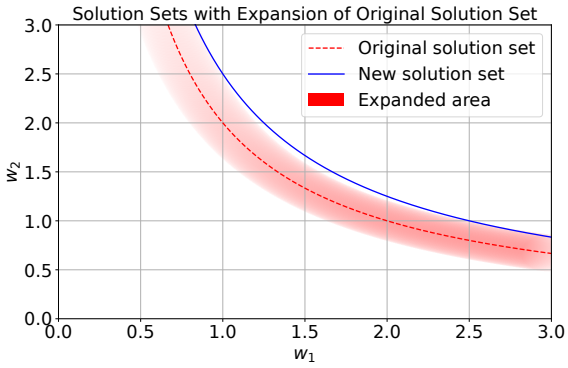


Figure 2: The location of the original solution set $S(\bar{x})$ and the new solution set $S(x^p)$. It presents how the $S(\bar{x})$ expands to achieve $S(x^p)$, where the red area is the expanded area of $S(\bar{x})$ as $S(\bar{x}) + \kappa \|x - x^p\| \mathbb{B}$.

Following Lemma 6 (see appendix B) and Theorem 1, the Lipschitz modulus κ for the toy model is 0.2. Given that the perturbation distance $\|x - x^p\|$ is known, we can deduce that $\kappa \|x - x^p\| \mathbb{B}$ is now a scaled unit ball with a radius of approximately 0.178. By definition in (4), the Lipschitz-like property reveals that the solution set after perturbation, i.e. $S(x^p)$, will fall into the expanded set $S(\bar{x}) + \kappa \|x - x^p\| \mathbb{B}$. We plot this fact in Figure 2, which shows that the expanded area of $S(\bar{x})$ is very close to and follows well with $S(x^p)$.

We estimate the new solution using $S(x^p) \approx \bar{w} + DS(\bar{x} \mid \bar{w})(\Delta x)$ (see (15)) and display five estimated solutions in $S(x^p) \cap \mathbb{B}(\bar{x}, 0.2)$ in Figure 3. As shown in Figure 3 (a), both the pristine and poisoned models have their respective set of solutions. The solution set of the poisoned model is shifted upwards compared to the original solution set. Our estimated solutions (the red points) are very close to the real solution set of the poisoned model. Figure 3 (b) demonstrates the loss landscape of the poisoned model. If we continue to use the original solution $\bar{w}$, the empirical risk would be 0.4. By shifting the solution from $\bar{w}$ to our estimated solutions, we can decrease the risk to nearly zero (around 0.01).

By estimating the new solution using the classical approach (see Eq. 3) and substituting the pseudo-inverse of the Hessian for the Hessian inverse, we can only derive a single estimated solution with a loss of 0.012 on the perturbed data, marked as the orange point in Figure (3). This solution is close to the estimated solution set derived using the graphical derivative, highlighting that the classical approach can be viewed as a special case of our proposed method.

Using Algorithm 1, the training stability can be measured by the minimum distance between $\bar{w}$ and the estimated solutions in $S(x^p) \cap \mathbb{B}(\bar{x}, 0.2)$ (e.g. the red points in Figure 3,)

which is 0.1746, representing the minimum point-to-point distance between the original solution and all estimated solutions. The experimental results are consistent with Figure 2, where the original solution set expands outward by $0.2 \|x - x^p\| \mathbb{B}$.

Simulation on Resnet

This section applies our estimated method, as defined by (15), to a Resnet56 network (He et al. 2016). We extract 1000 points from the CIFAR-10 dataset, denoted as $\bar{x}$. We use x^p to denote the poisoned data. The original solution $\bar{w}$ is the pre-trained weights of Resnet56.

Following (15), when the data $\bar{x}$ is perturbed along the direction Δx, the corresponding change direction of $\bar{w}$, denoted by Δw, is determined by the graphical derivative $DS(\bar{x} \mid \bar{w})(\Delta x)$. The relationship between Δx and Δw is:

$$\begin{aligned} &\nabla_w^2 \frac{1}{|I|} \sum_{i \in I} L\left(x_i, y_i, w\right) \Delta w \\ &+ \frac{1}{|K|} \sum_{i \in K} \nabla_{x_i} \nabla_w L\left(x_i, y_i, w\right) \Delta x_i = 0 \end{aligned} \tag{17}$$

Denoting † the pseudo inverse operator, Δw is given by:

$$\begin{aligned} \Delta w = &- \left(\nabla_w^2 \frac{1}{|I|} \sum_{i \in I} L\left(x_i, y_i, w\right) \right)^{\dagger} \\ &\left(\frac{1}{|K|} \sum_{i \in K} \nabla_{x_i} \nabla_w L\left(x_i, y_i, w\right) \Delta x_i \right). \end{aligned} \tag{18}$$

Note that equation (18) is also used in practical computations of (3), but it plays a different role in our paper. While other papers used the pseudo-inverse as a substitute for the Hessian inverse, we utilize it here to solve equation (17) and thereby estimate the graphical derivative following Theorem 1. To solve (18) under high-dimensional cases, we transform (18) to a least square problem:

$$\Delta w := \underset{v}{\operatorname{argmin}} \frac{1}{2} \left\| \begin{aligned} &\nabla_w^2 \frac{1}{|I|} \sum_{i \in I} L\left(x_i, y_i, w\right) v \\ &+ \frac{1}{|K|} \sum_{i \in K} \nabla_{x_i} \nabla_w L\left(x_i, y_i, w\right) \Delta x_i \end{aligned} \right\|^2 \tag{19}$$

where both $\nabla_x \nabla_w L\left(x_i, y_i, w\right) \Delta x$ and $\nabla_w^2 L\left(x_i, y_i, w\right) v$ can be calculated using implicit Hessian-vector products (HVP) (Pearlmutter 1994). As shown previously (Agarwal, Bullins, and Hazan 2017), $\nabla_w^2 L\left(x_i, y_i, w\right) v$ can be computed efficiently in $O(p)$.

We perturb an individual point x_k along the direction of $\nabla_{x_k} L(\bar{w})$. If we continue to use the $\bar{w}$ as our weights, the training loss for the points perturbed would increase from near zero to around 0.0041, due to the perturbations. This indicates that the original solution is far from the solution set of the poisoned model. We estimate the change of $\bar{w}$, i.e. Δw through solving the problem (19). By moving the original solution along Δw, we obtain the estimated solution. By adopting our estimated solutions, the training loss decreases to near zero, demonstrating that our estimated solution change Δw effectively draws $\bar{w}$ towards the solution set of the poisoned model. Table 1 reports the loss on the perturbed points for varying numbers of perturbed points when we adopt the original solution and the estimated solution. We then perturb 500 points and run the problem (19) with different random seeds, deriving different numerical results for Δw. Figure 4 demonstrates the location of the original solution and estimated solutions on the loss landscape of poisoned Resnet56. The loss landscape visualization follows the work in (Li et al. 2018), where the color intensity and the left plot's vertical axis indicate the loss. By moving the original solution $\bar{w}$ along Δw, the solutions reach the valley of the landscape of the poisoned model. Each estimated solution in Figure 4 corresponds to a simulation result of (19).

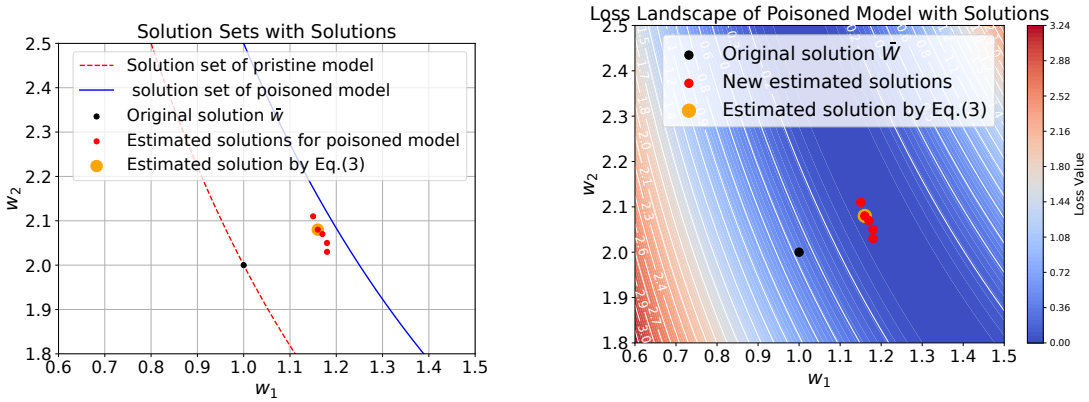
Solution \ perturbation	one point	10 points
Original Solution	0.0041	0.0045
Estimated Solution	1.9×10^{-5}	6.2×10^{-5}

Table 1: Empirical loss of the Resnet56 on the perturbed points using original and estimated solutions.

Related Work

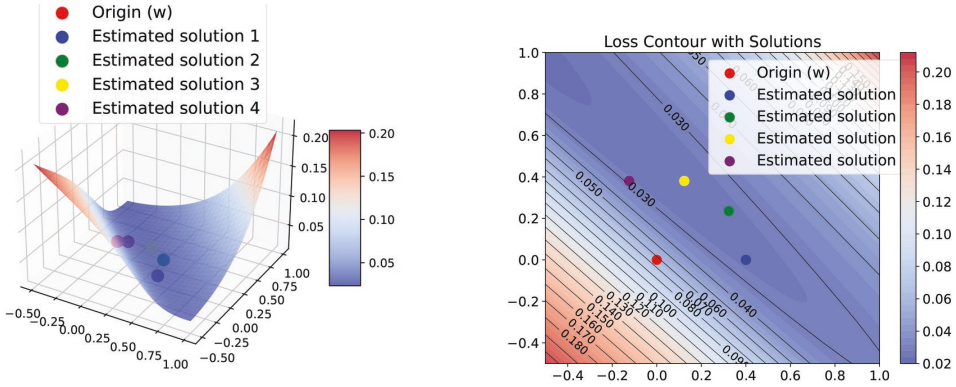
This paper only focuses on the sensitivity of solutions of learning models (e.g., model weights of DNNs) in response to perturbations in the training data. For the change of prediction in relationship to the inference data, one can refer to Fazlyab et al. (2019); Weng et al. (2018). Influence function, as a concept in robust statistics (Law 1986), was first used for the sensitivity analysis of the classification models with convex loss function (e.g. SVM, Logistic Regression) (Christmann and Steinwart 2004). Koh and Liang (2017) introduced it to the DNN, demonstrating its application in data poisoning attacks and identifying the important features. However, the existence of the influence function relies on the implicit function theorem (see Theorem 19 in Christmann and Steinwart (2004)), which may not be applicable to DNNs when non-isolated DNN solutions are considered, as discussed in the Introduction section of this paper. Nickl et al. (2024) measured the sensitivity of solutions to training data through the Memory-Perturbation Equation (MPE). It was demonstrated that sensitivity to a group of examples can be estimated by adding their natural gradients, indicating higher sensitivity with larger gradients. However, its theorem relies on the inverse of Hessian, which does not exist when the loss function is not strongly convex.

This paper utilizes the Lipschitz modulus to characterize the sensitivity of DNN. Noteworthy is that sensitivity analysis in our paper studies how the model *solution* changes with training data, not how the model *output* changes with inference data, although the two are closely related. To our best knowledge, this is the first time that the Lipschitz concept and the estimation of Lipschitz modulus estimation are



(a) (b)

Figure 3: (a) Respective solution set of the pristine and poisoned model. (b) The loss landscape of the poisoned toy model. The black,red points, orange point indicate the positions of the original solution $\bar{w}$, our estimated solutions and unique solution estimated by equation (3), respectively. The contour lines represent the corresponding losses.



(a) (b)

Figure 4: (a) 3D image and (b) 2D contours of the loss landscape of the poisoned ResNet56, indicating how the Δw draws the original solution towards the valley of the landscape.

introduced to the training stage. Previous research only focused on estimating the Lipschitz constants during the inference stage, see Fazlyab et al. (2019); Virmaux and Scaman (2018), to quantify the robustness of model prediction w.r.t. perturbations in inference data. For example, Virmaux and Scaman (2018) adopted a power method working with auto differentiation to estimate the upper bound of the Lipschitz constant. Most sensitivity analysis approaches only focus on a single solution of the learning algorithm. This paper considers the first-order optimality condition as a set-valued mapping (multifunction), introducing the 'set-to-set' analysis approach. Our sensitivity analysis is based on the Lipschitz-like property of set-valued mapping, where the Lipschitz Modulus quantifies the change of the solution set. For more discussion about set-valued mapping, interested readers can refer to Rockafellar and Wets (2009); Dontchev and Rockafellar (2009).

Conclusion

This paper provides set-valued analysis methods to study the sensitivity of model solutions (e.g. weights of a DNN) in response to perturbations in the training data. Theoretically, our approach considers the possibility that the DNN may not have unique solutions and does not rely on a non-singular Hessian. We accurately estimate the solution change when the training data are perturbed along a specific direction. Our analysis framework has multiple applications. It can leverage Lipschitz continuity to assess DNN sensitivity, where a larger Lipschitz constant indicates higher sensitivity. Our framework also extends the implicit function theorem for DNNs, enabling the execution of model target poisoning attacks by determining the perturbation direction to shift the solution toward a target (Suya et al. 2021).

Acknowledgments

The authors would like to express their gratitude to R. Tyrrell Rockafellar for his invaluable contributions and insights that greatly influenced this work.

References

Agarwal, N.; Bullins, B.; and Hazan, E. 2017. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116): 1–40.

Benton, G.; Maddox, W.; Lotfi, S.; and Wilson, A. G. G. 2021. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, 769–779. PMLR.

Christmann, A.; and Steinwart, I. 2004. On robustness properties of convex risk minimization methods for pattern recognition. *The Journal of Machine Learning Research*, 5: 1007–1034.

Cooper, Y. 2021. Global minima of overparameterized neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2): 676–691.

Dontchev, A. L.; and Rockafellar, R. T. 2009. *Implicit functions and solution mappings*, volume 543. Springer.

Fazlyab, M.; Robey, A.; Hassani, H.; Morari, M.; and Pappas, G. 2019. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in neural information processing systems*, 32.

Fiacco, A. V. 1983. Introduction to sensitivity and stability analysis in nonlinear programming. *(No Title)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, 1885–1894. PMLR.

Law, J. 1986. Robust statistics—the approach based on influence functions.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

Mei, S.; and Zhu, X. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the aaai conference on artificial intelligence*, volume 29.

Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E. C.; and Roli, F. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 27–38.

Nickl, P.; Xu, L.; Tailor, D.; Möllenhoff, T.; and Khan, M. E. E. 2024. The Memory-Perturbation Equation: Understanding Model's Sensitivity to Data. *Advances in Neural Information Processing Systems*, 36.

Pearlmutter, B. A. 1994. Fast exact multiplication by the Hessian. *Neural computation*, 6(1): 147–160.

Rockafellar, R. T.; and Wets, R. J.-B. 2009. *Variational analysis*, volume 317. Springer Science & Business Media.

Suya, F.; Mahloujifar, S.; Suri, A.; Evans, D.; and Tian, Y. 2021. Model-targeted poisoning attacks with provable convergence. In *International Conference on Machine Learning*, 10000–10010. PMLR.

Virmaux, A.; and Scaman, K. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31.

Weng, T.-W.; Zhang, H.; Chen, P.-Y.; Yi, J.; Su, D.; Gao, Y.; Hsieh, C.-J.; and Daniel, L. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*.

Yeung, D. S.; Cloete, I.; Shi, D.; and wY Ng, W. 2010. *Sensitivity analysis for neural networks*. Springer.