Sound and Complete Neural Network Repair with Minimality and Locality Guarantees

Reporter: Chi Zhiming

December 30, 2022

目录

目录

Contribution

1. The method is both sound and complete: the repaired network is guaranteed to fix the buggy input, and a patch is guaranteed to be found for any buggy input
2. This approach preserves the continuous piecewise linear nature of ReLU networks.
3. It can also automatically generalizes the repair to all the points including other undetected buggy inputs inside the repair region, is minimal in terms of changes in the function space, and guarantees that outputs on inputs away from the repair region are unaltered.

Contribution

1. The method is both sound and complete: the repaired network is guaranteed to fix the buggy input, and a patch is guaranteed to be found for any buggy input
2. This approach preserves the continuous piecewise linear nature of ReLU networks.
3. It can also automatically generalizes the repair to all the points including other undetected buggy inputs inside the repair region, is minimal in terms of changes in the function space, and guarantees that outputs on inputs away from the repair region are unaltered.

Contribution

1. The method is both sound and complete: the repaired network is guaranteed to fix the buggy input, and a patch is guaranteed to be found for any buggy input
2. This approach preserves the continuous piecewise linear nature of ReLU networks.
3. It can also automatically generalizes the repair to all the points including other undetected buggy inputs inside the repair region, is minimal in terms of changes in the function space, and guarantees that outputs on inputs away from the repair region are unaltered.

Existing network repair categories

- Retrainingfine-tuning:
  - Retrain or fine-tune the network with the newly identified buggy inputs and the corresponding corrected outputs
  - No guarantee, expensive and requires access to the original training data
- Direct weight modification:
  - optimization or a verification problem
  - optimization-based approach cannot guarantee removal of the buggy behaviors,
  - and the verification-based approach does not scale beyond networks of a few hundred neurons.
  - both approaches suffer from substantial accuracy drops on normal inputs.
- Architecture extension:
  - only work: PRDNN
  - The decoupling causes the repaired network to become discontinuous (in the functional sense).
  - It still cannot isolate the output change to a single buggy input from the rest of the

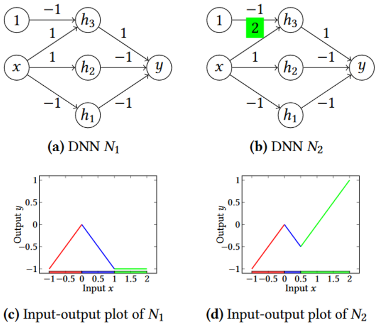
Existing network repair categories

- Retrainingfine-tuning:
  - Retrain or fine-tune the network with the newly identified buggy inputs and the corresponding corrected outputs
  - No guarantee, expensive and requires access to the original training data
- Direct weight modification:
  - optimization or a verification problem
  - optimization-based approach cannot guarantee removal of the buggy behaviors,
  - and the verification-based approach does not scale beyond networks of a few hundred neurons.
  - both approaches suffer from substantial accuracy drops on normal inputs.
- Architecture extension:
  - only work: PRDNN
  - The decoupling causes the repaired network to become discontinuous (in the functional sense).
  - It still cannot isolate the output change to a single buggy input from the rest of the

Existing network repair categories

- Retrainingfine-tuning:
  - Retrain or fine-tune the network with the newly identified buggy inputs and the corresponding corrected outputs
  - No guarantee, expensive and requires access to the original training data
- Direct weight modification:
  - optimization or a verification problem
  - optimization-based approach cannot guarantee removal of the buggy behaviors,
  - and the verification-based approach does not scale beyond networks of a few hundred neurons.
  - both approaches suffer from substantial accuracy drops on normal inputs.
- Architecture extension:
  - only work: PRDNN
  - The decoupling causes the repaired network to become discontinuous (in the functional sense).
  - It still cannot isolate the output change to a single buggy input from the rest of the

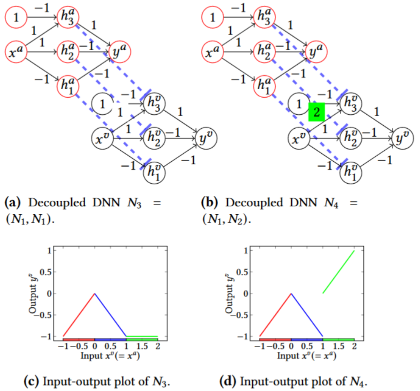
PRDNN



(a) DNN N_1 (b) DNN N_2

(c) Input-output plot of N_1 (d) Input-output plot of N_2

Figure 3. Example DNNs and their input-output behavior. The h_i nodes have ReLU activation functions. Colored bars on the x axis denote the linear regions.



(a) Decoupled DNN $N_3 = (N_1, N_1)$. **(b)** Decoupled DNN $N_4 = (N_1, N_2)$.

(c) Input-output plot of N_3. **(d)** Input-output plot of N_4.

Common Problem: Modification are global

1. This means that a correct behavior on another input, regardless of how far it is away from the buggy input, may not be preserved by the repair.
2. Worse still, the repair on a new buggy input can end up invalidating the repair on a previous buggy input.
3. Reason: It only poses a structural constraint, and does not limit the changes on the input-output mapping of the neural network.

Common Problem: Modification are global

1. This means that a correct behavior on another input, regardless of how far it is away from the buggy input, may not be preserved by the repair.
2. Worse still, the repair on a new buggy input can end up invalidating the repair on a previous buggy input.
3. Reason: It only poses a structural constraint, and does not limit the changes on the input-output mapping of the neural network.

Common Problem: Modification are global

1. This means that a correct behavior on another input, regardless of how far it is away from the buggy input, may not be preserved by the repair.
2. Worse still, the repair on a new buggy input can end up invalidating the repair on a previous buggy input.
3. Reason: It only poses a structural constraint, and does not limit the changes on the input-output mapping of the neural network.

Some Definition

- Linear Regions:
  - the set of inputs that correspond to the same activation pattern in a ReLU DNN f
  - a convex polytope, in the input space X on which f is linear.
- Correctness Specification:
  - $\Phi = (\Phi_{in}, \Phi_{out})$, Φ_{in} is the union of some linear regions and Φ_{out} is a convex polytope.
- Repair:
  - Area repair & Point-wise repair
  - Minimal repair: maximum distance between f and *repair*(f) over the whole input domain X

Some Definition

- Linear Regions:
    - the set of inputs that correspond to the same activation pattern in a ReLU DNN f
    - a convex polytope, in the input space X on which f is linear.
- Correctness Specification:
    - $\Phi = (\Phi_{in}, \Phi_{out})$, Φ_{in} is the union of some linear regions and Φ_{out} is a convex polytope.
- Repair:
    - Area repair & Point-wise repair
    - Minimal repair: maximum distance between f and *repair*(f) over the whole input domain X

Some Definition

- Linear Regions:
  - the set of inputs that correspond to the same activation pattern in a ReLU DNN f
  - a convex polytope, in the input space X on which f is linear.
- Correctness Specification:
  - $\Phi = (\Phi_{in}, \Phi_{out})$, Φ_{in} is the union of some linear regions and Φ_{out} is a convex polytope.
- Repair:
  - Area repair & Point-wise repair
  - Minimal repair: maximum distance between f and *repair*(f) over the whole input domain X

Repair Desiderata

1. Preservation of CPWL.
2. Sound and Complete.
3. Generalization.
4. Locality and Limited side effect.
5. Minimality: a significant departure from existing methods, that focus on minimizing the change in weights which has no guarantee on the amount of change in the function space.

Repair Desiderata

1. Preservation of CPWL.
2. Sound and Complete.
3. Generalization.
4. Locality and Limited side effect.
5. Minimality: a significant departure from existing methods, that focus on minimizing the change in weights which has no guarantee on the amount of change in the function space.

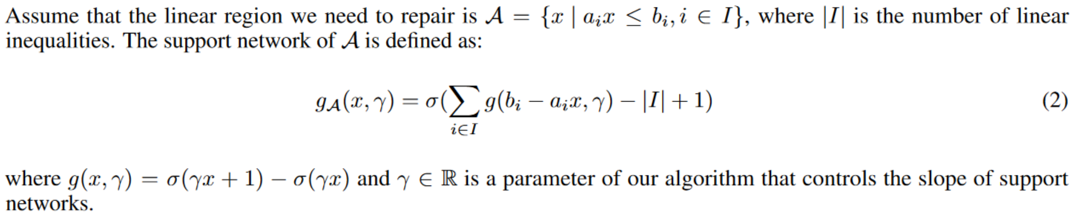
目录

Outline

- Given a linear region $\mathcal{A}$, The approach is to synthesize a patch network $h_{\mathcal{A}}$ such that $\widehat{f} = f + h_{\mathcal{A}}$ and $\widehat{f} \models \Phi$, which has a combination of two sub-networks.
- A support network $g_{\mathcal{A}}$ which behaves like a characteristic function to ensure that $h_{\mathcal{A}}$ is almost only active on $\mathcal{A}$.
- An affine patch function network $p_{\mathcal{A}}(x) = \boldsymbol{c}x + d$ such that $(f + p_{\mathcal{A}}) = \Phi$ on $\mathcal{A}$.

Support Networks

- This structure approximates the characteristic function, which are keys to ensuring localized repairs in our algorithm.

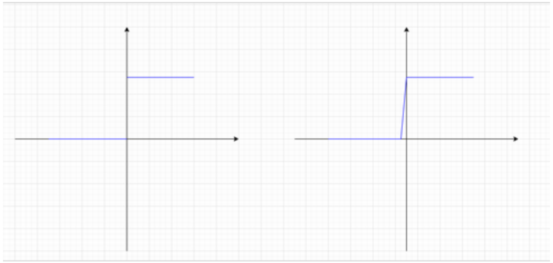


Assume that the linear region we need to repair is $\mathcal{A} = \{x \mid a_i x \le b_i, i \in I\}$, where $|I|$ is the number of linear inequalities. The support network of $\mathcal{A}$ is defined as:

$$g_{\mathcal{A}}(x, \gamma) = \sigma(\sum_{i \in I} g(b_i - a_i x, \gamma) - |I| + 1) \tag{2}$$

where $g(x, \gamma) = \sigma(\gamma x + 1) - \sigma(\gamma x)$ and $\gamma \in \mathbb{R}$ is a parameter of our algorithm that controls the slope of support networks.

Explanation of Support Networks

1. Discontinuous $\rightarrow$ Continuous:
2. 
3. For $g(x,\gamma) = \sigma(\gamma x + 1) - \sigma(\gamma x)$, if $0 > x \geq -\frac{1}{\gamma}$, then $g(x,\gamma) = \gamma x + 1$.

Explanation of Support Networks

- For $a_i x \leq b_i$, it has equivalent formula:
- $$\lim_{\gamma \to +\infty} \gamma(b_i - a_i x) + 1 \geq 0 \tag{1}$$
- And the γ is an adjustable parameter.

Affine Patch Functions

- To satisfy the desiderd property
- To obtain a minimal repair

$$\begin{cases} \min_{\boldsymbol{c},d} \max_{x \in \mathcal{A}} |p_{\mathcal{A}}(x)| = |\boldsymbol{c}x + d| \\ (\boldsymbol{c}, d) \in \{(\boldsymbol{c}, d) \mid f(x) + \boldsymbol{c}x + d \in \Phi_{\text{out}} \ , \forall x \in \mathcal{A}\} \end{cases} \tag{2}$$

- Vertices of polyhedron $\rightarrow$ expensive

$$\begin{cases} \min_{c,d} H \\ H \geq (cv_s + d)_i \, , H \geq -(cv_s + d)_i \, , \text{ for } s = 1, 2, \ldots, S \text{ and } i = 1, 2, \ldots, m \\ f(v_s) + p_{\mathcal{A}}(v_s) \in \Phi_{\text{out}} \ , \text{ for } s = 1, 2, \ldots, S \end{cases} \tag{3}$$

Affine Patch Functions

- To satisfy the desiderd property
- To obtain a minimal repair

$$\begin{cases} \min_{\boldsymbol{c},d} \max_{x\in\mathcal{A}} |p_{\mathcal{A}}(x)| = |\boldsymbol{c}x + d| \\ (\boldsymbol{c}, d) \in \{(\boldsymbol{c}, d) \mid f(x) + \boldsymbol{c}x + d \in \Phi_{\text{out}}\ , \forall x \in \mathcal{A}\} \end{cases} \tag{2}$$

- Vertices of polyhedron $\rightarrow$ expensive

$$\begin{cases} \min_{c,d} H \\ H \geq (cv_s + d)_i\,, H \geq -(cv_s + d)_i\,,\ \text{for } s = 1, 2, \ldots, S \text{ and } i = 1, 2, \ldots, m \\ f(v_s) + p_{\mathcal{A}}(v_s) \in \Phi_{\text{out}}\ ,\ \text{for } s = 1, 2, \ldots, S \end{cases} \tag{3}$$

Affine Patch Functions

- To satisfy the desiderd property
- To obtain a minimal repair

$$\begin{cases} \min_{\mathbf{c},d} \max_{x\in\mathcal{A}} |p_{\mathcal{A}}(x)| = |\mathbf{c}x + d| \\ (\mathbf{c}, d) \in \{(\mathbf{c}, d) \mid f(x) + \mathbf{c}x + d \in \Phi_{\text{out}} \ , \forall x \in \mathcal{A}\} \end{cases} \tag{2}$$

- Vertices of polyhedron $\rightarrow$ expensive

$$\begin{cases} \min_{c,d} H \\ H \geq (cv_s + d)_i \, , H \geq -(cv_s + d)_i \, , \ \text{for} \ s = 1, 2, \ldots, S \text{ and } i = 1, 2, \ldots, m \\ f(v_s) + p_{\mathcal{A}}(v_s) \in \Phi_{\text{out}} \ , \ \text{for} \ s = 1, 2, \ldots, S \end{cases} \tag{3}$$

Repair via Robust Optimization

- The optimization problem (3) can be converted to the following optimization problem, assuming $\Phi_{out} = \{y | a_{out} \cdot y \; b_{out}\}$:

$$\begin{cases} \min_{c,d} H \\ H \geq H_1, \text{ where } H_1 \text{ is the maximum value of the following inner } LP \begin{cases} \max_x cx + d \\ x \in \mathcal{A} \end{cases} \\ H \geq H_2, \text{ where } H_2 \text{ is the maximum value of the following inner } LP \begin{cases} \max_x -cx - d \\ x \in \mathcal{A} \end{cases} \\ b_{\text{out}} \geq H_3, \text{ where } H_3 \text{ is the maximum value of the following inner } LP \begin{cases} \max_x a_{\text{out}} \cdot (\\ x \in \mathcal{A} \end{cases} \end{cases} \tag{4}$$

Repair via Robust Optimization

- we can take the dual of inner LPs to avoid enumerating the vertices of $\mathcal{A}$

$$H_1 = \begin{cases} \max_x cx + d \\ x \in \mathcal{A} \end{cases} = d + \begin{cases} \max_x cx \\ ax \leq b \end{cases} \overset{\text{dual}}{=} d + \begin{cases} \min_p p'b \\ a'p = c \\ p \geq 0 \end{cases} \tag{5}$$

Repair via Robust Optimization

- This procedure is known as taking the *robust counterpart* Ben-Tal et al. [2009] of the original problem.

$$\begin{cases} \min_{\boldsymbol{c},d,p_1,p_2,q,H} H \\ H \geq p_1' b + d,\ a' p_1 = \boldsymbol{c},\ p_1 \geq 0 \\ H \geq p_2' b - d,\ a' p_2 = -\boldsymbol{c},\ p_2 \geq 0 \\ b_{\text{out}} \geq q' b + a_{\text{out}} \left(d_f + d\right),\ a' q = a_{\text{out}} \left(\boldsymbol{c_f} + \boldsymbol{c}\right),\ q \geq 0 \end{cases} \tag{6}$$

where $f(x) = \boldsymbol{c_f} \cdot x + d_f$.

Single-Region Repairs

With a support network $g_{\mathcal{A}}$ and an affine patch function $p_{\mathcal{A}}$, we can synthesize the final patch network as follows:

$$h_{\mathcal{A}}(x,\gamma) = \sigma(p_{\mathcal{A}}(x) + K \cdot g_{\mathcal{A}}(x,\gamma) - K) - \sigma(-p_{\mathcal{A}}(x) + K \cdot g_{\mathcal{A}}(x,\gamma) - K) \tag{10}$$

where K is a vector with every entry is equal to the upper bound of $\{|p_{\mathcal{A}}(x)|_{+\infty} | x \in X\}$.

Remark: For $x \in \mathcal{A}$, $g_{\mathcal{A}}(x,\gamma) = 1$, then we have $h_{\mathcal{A}}(x,\gamma) = \sigma(p_{\mathcal{A}}(x)) - \sigma(-p_{\mathcal{A}}(x)) = p_{\mathcal{A}}(x)$. For $x \notin \mathcal{A}$, $g_{\mathcal{A}}(x,\gamma)$ goes to zero quickly if γ is large. When $g_{\mathcal{A}}(x,\gamma) = 0$, we have $h_{\mathcal{A}}(x,\gamma) = \sigma(p_{\mathcal{A}}(x) - K) - \sigma(-p_{\mathcal{A}}(x) - K) = 0$.

The repaired network $\widehat{f}(x) = f(x) + h_{\mathcal{A}}(x,\gamma)$. Since f and $h_{\mathcal{A}}$ are both ReLU DNNs, we have $\widehat{f}$ is also a ReLU DNN. We will give the formal guarantees on correctness in Theorem 1.

Multi-Region Repairs

Observations:

- If Linregion $\mathcal{A}_1 \cap \mathcal{A}_2 \neq \emptyset$, for any $x \in \mathcal{A}_1 \cap \mathcal{A}_2$, both $h_{\mathcal{A}_1}(x, \gamma)$ and $h_{\mathcal{A}_2}(x, \gamma)$ will alter the value of f on x, which will invalidate both repairs and cannot guarantee that the repaired DNN will meet the specification Φ.
- For multi-region repair, we note $\{\mathcal{A}_l\}_{l=1,2,\ldots,L}$ are all the buggy linear regions.
  1. We compute the support network $g_{\mathcal{A}_i}(x, \gamma)$ and affine patch function $p_{\mathcal{A}_i}(x)$ for each A_i in parallel.
  2. We repair $\mathcal{A}_i \cup \cdots \cup \mathcal{A}_L$ with $p_{\mathcal{A}_i}(x) - p_{\mathcal{A}_{i-1}}(x)$ in turn.
  3. We use $\max_{j \geq i}\{g_{\mathcal{A}_j}(x, \gamma)\}$ as characteristic function of $\mathcal{A}_i \cup \cdots \cup \mathcal{A}_L$

Multi-Region Repairs

Observations:

- If Linregion $\mathcal{A}_1 \cap \mathcal{A}_2 \neq \emptyset$, for any $x \in \mathcal{A}_1 \cap \mathcal{A}_2$, both $h_{\mathcal{A}_1}(x, \gamma)$ and $h_{\mathcal{A}_2}(x, \gamma)$ will alter the value of f on x, which will invalidate both repairs and cannot guarantee that the repaired DNN will meet the specification Φ.
- For multi-region repair, we note $\{\mathcal{A}_l\}_{l=1,2,\ldots,L}$ are all the buggy linear regions.
  1. We compute the support network $g_{\mathcal{A}_i}(x, \gamma)$ and affine patch function $p_{\mathcal{A}_i}(x)$ for each A_i in parallel.
  2. We repair $\mathcal{A}_i \cup \cdots \cup \mathcal{A}_L$ with $p_{\mathcal{A}_i}(x) - p_{\mathcal{A}_{i-1}}(x)$ in turn.
  3. We use $\max_{j \geq i} \left\{ g_{\mathcal{A}_j}(x, \gamma) \right\}$ as characteristic function of $\mathcal{A}_i \cup \cdots \cup \mathcal{A}_L$

An example

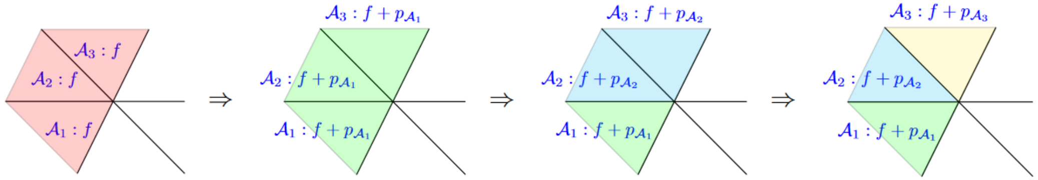

$\mathcal{A}_3 : f$
$\mathcal{A}_2 : f$
$\mathcal{A}_1 : f$
$\Rightarrow$
$\mathcal{A}_3 : f + p_{\mathcal{A}_1}$
$\mathcal{A}_2 : f + p_{\mathcal{A}_1}$
$\mathcal{A}_1 : f + p_{\mathcal{A}_1}$
$\Rightarrow$
$\mathcal{A}_3 : f + p_{\mathcal{A}_2}$
$\mathcal{A}_2 : f + p_{\mathcal{A}_2}$
$\mathcal{A}_1 : f + p_{\mathcal{A}_1}$
$\Rightarrow$
$\mathcal{A}_3 : f + p_{\mathcal{A}_3}$
$\mathcal{A}_2 : f + p_{\mathcal{A}_2}$
$\mathcal{A}_1 : f + p_{\mathcal{A}_1}$

Figure 3: An illustration of multi-region repair with three different repair regions. Left: the original DNN; Middle Left: repair $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ with $p_{\mathcal{A}_1}$; Middle Right: repair $\mathcal{A}_2 \cup \mathcal{A}_3$ with $p_{\mathcal{A}_2} - p_{\mathcal{A}_1}$; Right: repair $\mathcal{A}_3$ with $p_{\mathcal{A}_3} - p_{\mathcal{A}_2}$

Patch network of Multi-Region Repairs

$$\begin{aligned} h(x,\gamma) = \sum_{l} \Bigg[& \sigma\left(p_{\mathcal{A}_l}(x) - p_{\mathcal{A}_{l-1}}(x) + \max_{j\geq l}\left\{ g_{\mathcal{A}_j}(x,\gamma)\right\} K_l - K_l \right) \\ & -\sigma\left(-p_{\mathcal{A}_l}(x) + p_{\mathcal{A}_{l-1}}(x) + \max_{j\geq l}\left\{ g_{\mathcal{A}_j}(x,\gamma)\right\} K_l - K_l \right) \Bigg] \end{aligned} \tag{7}$$

where K_l is the upper bound of $\left\{ \left| p_{\mathcal{A}_l}(x) - p_{\mathcal{A}_{l-1}}(x) \right|_{\infty} \mid x \in X \right\}$ and $p_{\mathcal{A}_0}(x) = 0$.

Feature-Space Repairs

- For large DNN, the huge number of linear constraints for one patch area $\mathcal{A}$ will pose a challenge to solving the resulting LP.
- We can construct a patch network starting from an intermediate layer.
  - Divide DNN f into f_1 and f_2, i.e. $f = f_2 \circ f_1$.
  - Add a patch network from the output space of f_1, which is a **feature space**.
- To repair the network in feature space, we can
  - reduce the parameter overhead of the additional networks,
  - and make the repair process more computation-friendly.
- We loses the locality guarantee in the input space but still preserves **locality in the feature space**.

Feature-Space Repairs

- For large DNN, the huge number of linear constraints for one patch area $\mathcal{A}$ will pose a challenge to solving the resulting LP.
- We can construct a patch network starting from an intermediate layer.
  - Divide DNN f into f_1 and f_2, i.e. $f = f_2 \circ f_1$.
  - Add a patch network from the output space of f_1, which is a **feature space**.
- To repair the network in feature space, we can
  - reduce the parameter overhead of the additional networks,
  - and make the repair process more computation-friendly.
- We loses the locality guarantee in the input space but still preserves **locality in the feature space**.

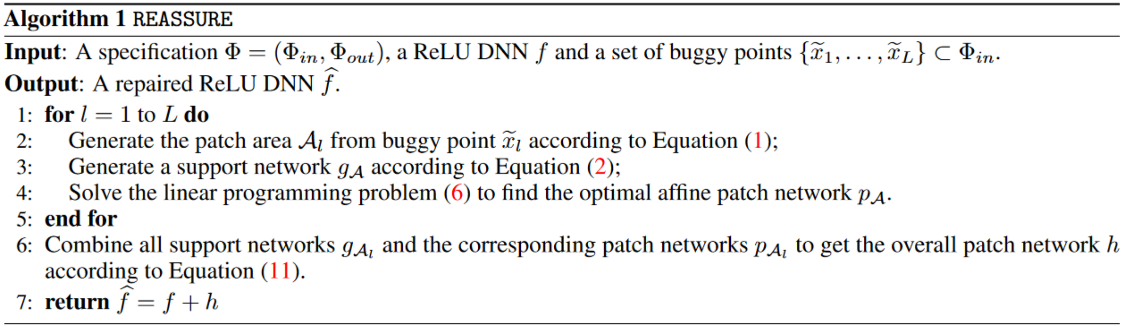
Feature-Space Repairs

- For large DNN, the huge number of linear constraints for one patch area $\mathcal{A}$ will pose a challenge to solving the resulting LP.
- We can construct a patch network starting from an intermediate layer.
  - Divide DNN f into f_1 and f_2, i.e. $f = f_2 \circ f_1$.
  - Add a patch network from the output space of f_1, which is a **feature space**.
- To repair the network in feature space, we can
  - reduce the parameter overhead of the additional networks,
  - and make the repair process more computation-friendly.
- We loses the locality guarantee in the input space but still preserves **locality in the feature space**.

Feature-Space Repairs

- For large DNN, the huge number of linear constraints for one patch area $\mathcal{A}$ will pose a challenge to solving the resulting LP.
- We can construct a patch network starting from an intermediate layer.
  - Divide DNN f into f_1 and f_2, i.e. $f = f_2 \circ f_1$.
  - Add a patch network from the output space of f_1, which is a **feature space**.
- To repair the network in feature space, we can
  - reduce the parameter overhead of the additional networks,
  - and make the repair process more computation-friendly.
- We loses the locality guarantee in the input space but still preserves **locality in the feature space**.

Algorithm



Algorithm 1 REASSURE

Input: A specification $\Phi = (\Phi_{in}, \Phi_{out})$, a ReLU DNN f and a set of buggy points $\{\widetilde{x}_1, \ldots, \widetilde{x}_L\} \subset \Phi_{in}$.

Output: A repaired ReLU DNN $\widehat{f}$.

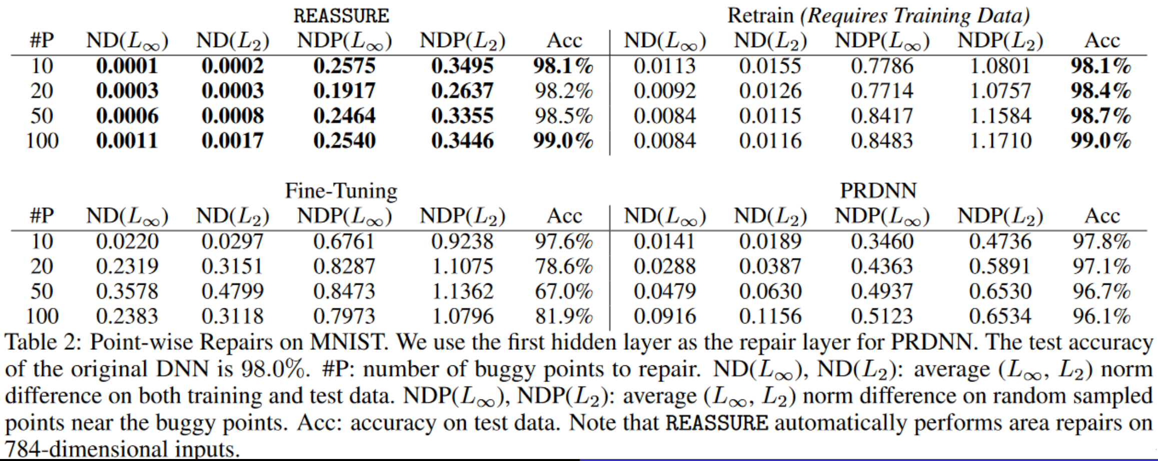
```
1: for $l = 1$ to $L$ do
2:     Generate the patch area $\mathcal{A}_l$ from buggy point $\widetilde{x}_l$ according to Equation (1);
3:     Generate a support network $g_{\mathcal{A}}$ according to Equation (2);
4:     Solve the linear programming problem (6) to find the optimal affine patch network $p_{\mathcal{A}}$.
5: end for
6: Combine all support networks $g_{\mathcal{A}_l}$ and the corresponding patch networks $p_{\mathcal{A}_l}$ to get the overall patch network $h$ according to Equation (11).
7: return $\widehat{f} = f + h$
```

目录

Notion

- **Q1** **Effectiveness**: How effective is a repair in removing known buggy behaviors?
- **Q2** **Locality**: How much side effect (i.e. modification outside the patch area in the function space) does a repair produce?
- **Q3** **Function Change**: How much does a repair change the original neural network in the function space?
- **Q4** **Performance**: Whether and how much does a repair adversely affect the overall performance of the neural network?

Point-wise Repairs: MNIST

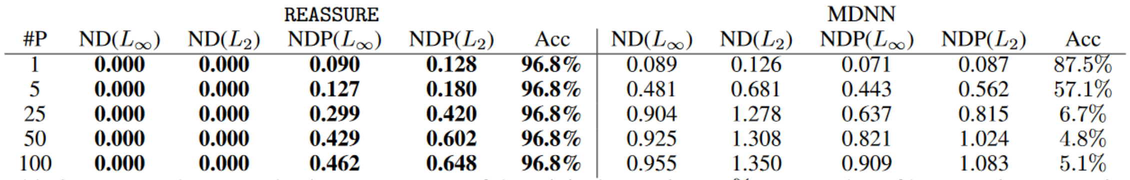


	REASSURE					Retrain *(Requires Training Data)*				
#P	ND(L_∞)	ND(L_2)	NDP(L_∞)	NDP(L_2)	Acc	ND(L_∞)	ND(L_2)	NDP(L_∞)	NDP(L_2)	Acc
10	**0.0001**	**0.0002**	**0.2575**	**0.3495**	**98.1%**	0.0113	0.0155	0.7786	1.0801	**98.1%**
20	**0.0003**	**0.0003**	**0.1917**	**0.2637**	98.2%	0.0092	0.0126	0.7714	1.0757	**98.4%**
50	**0.0006**	**0.0008**	**0.2464**	**0.3355**	98.5%	0.0084	0.0115	0.8417	1.1584	**98.7%**
100	**0.0011**	**0.0017**	**0.2540**	**0.3446**	**99.0%**	0.0084	0.0116	0.8483	1.1710	**99.0%**

	Fine-Tuning					PRDNN				
#P	ND(L_∞)	ND(L_2)	NDP(L_∞)	NDP(L_2)	Acc	ND(L_∞)	ND(L_2)	NDP(L_∞)	NDP(L_2)	Acc
10	0.0220	0.0297	0.6761	0.9238	97.6%	0.0141	0.0189	0.3460	0.4736	97.8%
20	0.2319	0.3151	0.8287	1.1075	78.6%	0.0288	0.0387	0.4363	0.5891	97.1%
50	0.3578	0.4799	0.8473	1.1362	67.0%	0.0479	0.0630	0.4937	0.6530	96.7%
100	0.2383	0.3118	0.7973	1.0796	81.9%	0.0916	0.1156	0.5123	0.6534	96.1%

Table 2: Point-wise Repairs on MNIST. We use the first hidden layer as the repair layer for PRDNN. The test accuracy of the original DNN is 98.0%. #P: number of buggy points to repair. ND(L_∞), ND(L_2): average (L_∞, L_2) norm difference on both training and test data. NDP(L_∞), NDP(L_2): average (L_∞, L_2) norm difference on random sampled points near the buggy points. Acc: accuracy on test data. Note that REASSURE automatically performs area repairs on 784-dimensional inputs.

Point-wise Repairs: MNIST



	REASSURE					MDNN				
#P	ND(L_∞)	ND(L_2)	NDP(L_∞)	NDP(L_2)	Acc	ND(L_∞)	ND(L_2)	NDP(L_∞)	NDP(L_2)	Acc
1	**0.000**	**0.000**	**0.090**	**0.128**	**96.8%**	0.089	0.126	0.071	0.087	87.5%
5	**0.000**	**0.000**	**0.127**	**0.180**	**96.8%**	0.481	0.681	0.443	0.562	57.1%
25	**0.000**	**0.000**	**0.299**	**0.420**	**96.8%**	0.904	1.278	0.637	0.815	6.7%
50	**0.000**	**0.000**	**0.429**	**0.602**	**96.8%**	0.925	1.308	0.821	1.024	4.8%
100	**0.000**	**0.000**	**0.462**	**0.648**	**96.8%**	0.955	1.350	0.909	1.083	5.1%

Table 3: Watermark Removal. The test accuracy of the original DNN is 96.8%. #P: number of buggy points to repair; ND(L_∞), ND(L_2): average (L_∞, L_2) norm difference on both training data and testing data; NDP(L_∞), NDP(L_2): average (L_∞, L_2) norm difference on random sampled points near watermark images; Acc: accuracy on test data.

Area Repairs: HCAS

- Only compare with PRDNN
- HCAS networks (simplified version of ACAS Xu)
  - Each of them has an input layer with 3 nodes, 5 hidden layers with 25 nodes in each hidden layer, and a final output layer with 5 nodes.
- We use the method from Girard-Satabin et al. [2021] to compute all the linear regions for the network(87 buggy linear regions were found.)
- We use Specification 1, which is similar to Property 5.

Area Repairs: HCAS

- Only compare with PRDNN
- HCAS networks (simplified version of ACAS Xu)
    - Each of them has an input layer with 3 nodes, 5 hidden layers with 25 nodes in each hidden layer, and a final output layer with 5 nodes.
- We use the method from Girard-Satabin et al. [2021] to compute all the linear regions for the network(87 buggy linear regions were found.)
- We use Specification 1, which is similar to Property 5.

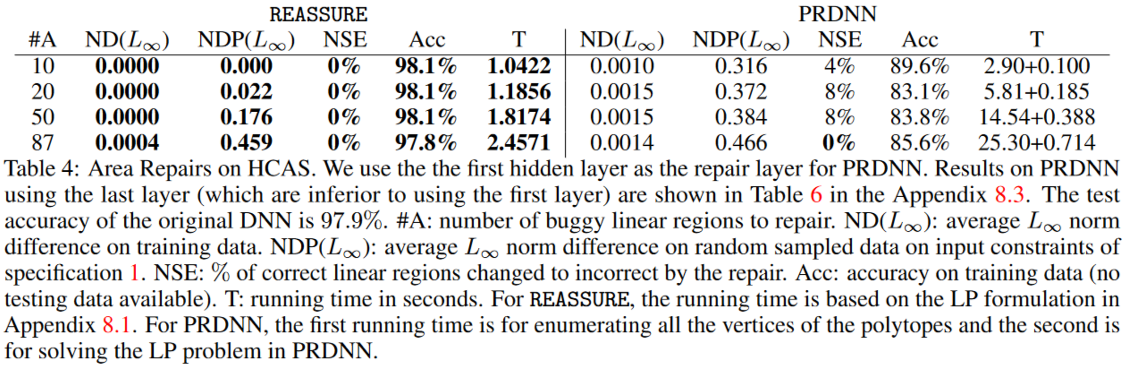
Area Repairs: HCAS

- Only compare with PRDNN
- HCAS networks (simplified version of ACAS Xu)
  - Each of them has an input layer with 3 nodes, 5 hidden layers with 25 nodes in each hidden layer, and a final output layer with 5 nodes.
- We use the method from Girard-Satabin et al. [2021] to compute all the linear regions for the network(87 buggy linear regions were found.)
- We use Specification 1, which is similar to Property 5.

Area Repairs: HCAS

- Only compare with PRDNN
- HCAS networks (simplified version of ACAS Xu)
  - Each of them has an input layer with 3 nodes, 5 hidden layers with 25 nodes in each hidden layer, and a final output layer with 5 nodes.
- We use the method from Girard-Satabin et al. [2021] to compute all the linear regions for the network(87 buggy linear regions were found.)
- We use Specification 1, which is similar to Property 5.

Area Repairs: HCAS



	REASSURE					PRDNN				
#A	ND(L_∞)	NDP(L_∞)	NSE	Acc	T	ND(L_∞)	NDP(L_∞)	NSE	Acc	T
10	**0.0000**	**0.000**	**0%**	**98.1%**	**1.0422**	0.0010	0.316	4%	89.6%	2.90+0.100
20	**0.0000**	**0.022**	**0%**	**98.1%**	**1.1856**	0.0015	0.372	8%	83.1%	5.81+0.185
50	**0.0000**	**0.176**	**0%**	**98.1%**	**1.8174**	0.0015	0.384	8%	83.8%	14.54+0.388
87	**0.0004**	**0.459**	**0%**	**97.8%**	**2.4571**	0.0014	0.466	**0%**	85.6%	25.30+0.714

Table 4: Area Repairs on HCAS. We use the the first hidden layer as the repair layer for PRDNN. Results on PRDNN using the last layer (which are inferior to using the first layer) are shown in Table 6 in the Appendix 8.3. The test accuracy of the original DNN is 97.9%. #A: number of buggy linear regions to repair. ND(L_∞): average L_∞ norm difference on training data. NDP(L_∞): average L_∞ norm difference on random sampled data on input constraints of specification 1. NSE: % of correct linear regions changed to incorrect by the repair. Acc: accuracy on training data (no testing data available). T: running time in seconds. For REASSURE, the running time is based on the LP formulation in Appendix 8.1. For PRDNN, the first running time is for enumerating all the vertices of the polytopes and the second is for solving the LP problem in PRDNN.

Feature-Space Repairs: ImageNet

- Only compare with PRDNN
- They only consider 10 output classes(found on ImageNet-A Hendrycks et al. [2021])
- They slightly modified AlexNet by a multilayer perceptron with three hidden layers to simplify the evaluation.
- They construct the patch network starting from the third from the last hidden layers.

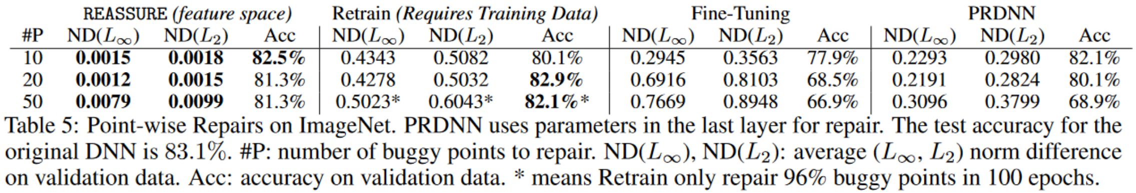
Feature-Space Repairs: ImageNet

- Only compare with PRDNN
- They only consider 10 output classes(found on ImageNet-A Hendrycks et al. [2021])
- They slightly modified AlexNet by a multilayer perceptron with three hidden layers to simplify the evaluation.
- They construct the patch network starting from the third from the last hidden layers.

Feature-Space Repairs: ImageNet

- Only compare with PRDNN
- They only consider 10 output classes(found on ImageNet-A Hendrycks et al. [2021])
- They slightly modified AlexNet by a multilayer perceptron with three hidden layers to simplify the evaluation.
- They construct the patch network starting from the third from the last hidden layers.

Feature-Space Repairs: ImageNet



	REASSURE *(feature space)*			Retrain *(Requires Training Data)*			Fine-Tuning			PRDNN		
#P	ND(L_∞)	ND(L_2)	Acc	ND(L_∞)	ND(L_2)	Acc	ND(L_∞)	ND(L_2)	Acc	ND(L_∞)	ND(L_2)	Acc
10	**0.0015**	**0.0018**	**82.5%**	0.4343	0.5082	80.1%	0.2945	0.3563	77.9%	0.2293	0.2980	82.1%
20	**0.0012**	**0.0015**	81.3%	0.4278	0.5032	**82.9%**	0.6916	0.8103	68.5%	0.2191	0.2824	80.1%
50	**0.0079**	**0.0099**	81.3%	0.5023*	0.6043*	**82.1%***	0.7669	0.8948	66.9%	0.3096	0.3799	68.9%

Table 5: Point-wise Repairs on ImageNet. PRDNN uses parameters in the last layer for repair. The test accuracy for the original DNN is 83.1%. #P: number of buggy points to repair. ND(L_∞), ND(L_2): average (L_∞, L_2) norm difference on validation data. Acc: accuracy on validation data. * means Retrain only repair 96% buggy points in 100 epochs.

目录

Parameter Overhead

- REASSURE introduces new parameters for per repair region is in O(m|I|)
- Use LP to check if a constraint is redundant.
  - For HCAS, only 3% of the original 125 constraints after removing the redundant constraints
- Use feature-space repairs instead of input space
  - 0.1%(500k) of parameters from input space.
  - trade off locality and parameter overhead.
  - When the number of points or linear regions to repair becomes **large**, it may make sense to perform repairs in the feature space anyway for **better generalization**.

Parameter Overhead

- REASSURE introduces new parameters for per repair region is in O(m|I|)
- Use LP to check if a constraint is redundant.
  - For HCAS, only 3% of the original 125 constraints after removing the redundant constraints
- Use feature-space repairs instead of input space
  - 0.1%(500k) of parameters from input space.
  - trade off locality and parameter overhead.
  - When the number of points or linear regions to repair becomes **large**, it may make sense to perform repairs in the feature space anyway for **better generalization**.

Parameter Overhead

- REASSURE introduces new parameters for per repair region is in O(m|I|)
- Use LP to check if a constraint is redundant.
    - For HCAS, only 3% of the original 125 constraints after removing the redundant constraints
- Use feature-space repairs instead of input space
    - 0.1%(500k) of parameters from input space.
    - trade off locality and parameter overhead.
    - When the number of points or linear regions to repair becomes **large**, it may make sense to perform repairs in the feature space anyway for **better generalization**.

Thank you