BTP492
Repair of Deep Neural Networks

Yash Bansal (2022CS51133)

Supervisors:
Prof. Priyanka Golia
Prof. Kumar Madhukar

November 27, 2025

1 Introduction

Deep Neural Networks (DNNs) are used in safety-critical applications like healthcare and autonomous driving. Even high-performing models can produce wrong results on specific inputs, failing to satisfy required properties.

Retraining from scratch is expensive, offers no guarantee of satisfying the required property, and may introduce new undesired behaviors.

This project addresses *DNN repair*: given a trained DNN N, a violated property Q, and counterexamples $X = \{x_1, x_2, \ldots, x_n\}$, construct a repaired network N' that satisfies Q on all counterexamples while preserving performance on other inputs.

Our approach uses gradient-based optimization with layer-wise modifications, leveraging the observation that changes in hidden layer outputs are proportional to their gradients. This enables minimal repairs without full network optimization at each step.

2 Problem Specification

2.1 DNN Verification Problem

Given a trained DNN N with parameters θ and a property Q, the verification problem asks: does N satisfy Q for all possible inputs?

Formally, we check the satisfiability of:

$$N(\theta) \land \neg Q \tag{1}$$

This can have two outcomes:

- **UNSAT**: The property Q holds for all inputs. The network is safe.
- **SAT**: The property is violated. The solver returns counterexamples $X = \{x_1, x_2, \ldots, x_n\}$ where $N(x_i)$ does not satisfy Q.

2.2 DNN Repair Problem

Given:

- A trained DNN N with weights $W = \{w_1, w_2, \ldots, w_m\}$
- A property Q that N violates
- A set of counterexamples $X = \{x_1, x_2, \ldots, x_n\}$

Find a repaired network N' with weights $W' = \{w'_1, w'_2, \ldots, w'_m\}$ such that:

1. **Correctness**: For all $x \in X$, $N'(x)$ satisfies property Q
2. **Minimality**: The distance between N and N' is minimized

2.3 Distance Metric

We define the distance between networks using the L_p norm:

$$\text{Distance}_L(N, N') = \left(\sum_{i=1}^{m} |w_i - w'_i|^L \right)^{1/L} \tag{2}$$

where $L = 1$ gives Manhattan distance, $L = 2$ gives Euclidean distance, and $L = \infty$ gives maximum absolute difference.

2.4 Optimization Formulation

The repair problem can be formulated as a constrained optimization:

$$\begin{aligned} \min_{W'} \quad & \text{Distance}_L(W, W') \\ \text{subject to} \quad & N'(x_i) \models Q, \quad \forall x_i \in X \end{aligned} \tag{3}$$

The constraint ensures that the repaired network satisfies the property on all counterexamples. Ideally, we also want to preserve the network's behavior on non-violating inputs.

2.5 Hypothesis: Weight Changes and Loss Preservation

We hypothesize that minimal weight modifications lead to minimal changes in the loss function for non-violating inputs. This suggests that by minimizing weight changes, we can fix property violations while keeping the network's performance intact on correctly classified inputs.

2.6 Challenge

The main challenge is that this optimization problem is high-dimensional, non-convex, constrained, and computationally expensive. Our goal is to develop methods that can solve this problem efficiently while finding minimal modifications to the network weights.

3 Initial Exploration and Approach

3.1 Understanding the Foundation

We studied the Veristable solver framework, DPLL(T) verification, and neuron stability concepts to understand how DNN verification works and how counterexamples are generated.

3.2 Solver Analysis and Exploration

We initially explored using the DPLL search tree from Veristable to guide repair:

- Debugged solver and analyzed counterexample traces to identify problematic weights
- Tracked decision paths and conflicts leading to property violations
- Attempted to use DPLL tree structure to target specific neurons for repair

Outcome: The DPLL tree did not reveal clear patterns for determining optimal weight modifications. Decision paths varied too much across counterexamples to extract consistent repair strategies.

We transitioned to an optimization-based approach for more direct computation of weight changes.

4 Final Approach

After exploring solver-based methods, we developed an optimization-based approach that combines gradient information with layer-wise modifications to achieve efficient and minimal DNN repair.

4.1 Overview

Our approach consists of three main steps for each counterexample:

1. Compute the nearest correct output that satisfies the property
2. Calculate gradients to identify repair directions
3. Apply layer-wise modifications recursively to propagate repairs

4.2 Computing Target Output

For a counterexample x_i with current output $\hat{y}_i = N(x_i)$ that violates property Q, we find the nearest output y_i^* that satisfies Q by solving:

$$\begin{aligned} \min_{y^*} \quad & \|y^* - \hat{y}_i\|^2 \\ \text{subject to} \quad & y^* \models Q \end{aligned} \tag{4}$$

This gives us a target output that is minimally different from the current incorrect output while satisfying the required property. We define the loss for this counterexample as:

$$\mathcal{L}_i = \|N(x_i) - y_i^*\|^2 \tag{5}$$

4.3 Gradient-Based Repair Direction

We compute gradients of the loss with respect to the pre-activation values at each hidden layer l:

$$\nabla_{z^{(l)}} \mathcal{L}_i = \frac{\partial \mathcal{L}_i}{\partial z^{(l)}} \tag{6}$$

where $z^{(l)}$ is the pre-activation output of layer l (before applying the activation function).

Key Observation: We found that the change needed in layer activations is approximately proportional to their gradients. This means we can estimate the required perturbation as:

$$\Delta z^{(l)} = -\alpha \cdot \nabla_{z^{(l)}} \mathcal{L}_i \tag{7}$$

where α is a step size parameter controlling the magnitude of the repair.

4.4 Layer-wise Modification

Instead of modifying all network weights simultaneously (which is computationally expensive), we repair one layer at a time. For a given layer l, we solve a constrained optimization problem to find minimal weight changes that achieve the desired activation perturbation.

4.4.1 Single Layer Repair Problem

Consider repairing layer l to achieve perturbed activations $z^{(l)} + \Delta z^{(l)}$. We formulate this as:

$$\begin{aligned} \min_{\Delta W^{(l)}} \quad & \|\Delta W^{(l)}\|_\infty \\ \text{subject to} \quad & (W^{(l)} + \Delta W^{(l)}) \cdot a^{(l-1)} = z^{(l)} + \Delta z^{(l)} \end{aligned} \tag{8}$$

where $W^{(l)}$ are the original weights, $\Delta W^{(l)}$ are the weight changes, and $a^{(l-1)}$ is the activation from the previous layer (fixed).

Why Layer-wise Modification is Fast: As shown in [1], this single-layer repair problem can be reformulated as a DNN verification problem. Since efficient DNN verification solvers already exist, we can leverage these solvers to find optimal weight modifications quickly.

4.5 Recursive Layer-wise Repair

The key insight from [2] is that we can apply layer-wise modifications recursively. The paper shows that modifying one layer at a time and propagating these changes through subsequent layers still achieves the global repair objective while maintaining minimality guarantees.

We apply layer-wise modifications recursively from a chosen layer towards both the input and output ends:

1. **Choose a middle layer l** based on gradient magnitude or other heuristics
2. **Backward repair (layers before l):**
   - Compute target activations for layer l: $z^{(l)}_{\text{target}} = z^{(l)} + \Delta z^{(l)}$
   - Solve optimization to modify weights in layers 1 to $l-1$ to achieve $z^{(l)}_{\text{target}}$
3. **Forward repair (layers after l):**
   - Use perturbed activations $a^{(l)} = \text{ReLU}(z^{(l)}_{\text{target}})$ as input
   - Solve optimization to modify weights in layers $l+1$ to L to achieve target output y^*
4. **Combine modifications:** Apply all weight changes to get repaired network N'

As demonstrated in [2], this recursive approach decomposes the global repair problem into smaller, more tractable subproblems.

4.6 Algorithm Summary

Algorithm 1 Gradient-Based Layer-wise DNN Repair

```
Require: Network $N$, counterexample $x$, property $Q$
Ensure: Repaired network $N'$
 1: $\hat{y} \leftarrow N(x)$
 2: $y^* \leftarrow \arg\min_{y:y\models Q} \|y - \hat{y}\|^2$
 3: $\mathcal{L} \leftarrow \|y^* - \hat{y}\|^2$
 4: for each hidden layer $l$ do
 5:     $\nabla_{z^{(l)}}\mathcal{L} \leftarrow \frac{\partial \mathcal{L}}{\partial z^{(l)}}$
 6: end for
 7: for each $\alpha$ in $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ do
 8:     for each hidden layer $l$ do
 9:         $\Delta z^{(l)} \leftarrow -\alpha \cdot \nabla_{z^{(l)}}\mathcal{L}$
10:     end for
11:     $l_r \leftarrow$ select repair layer
12:     $z^{(l_r)}_{\text{target}} \leftarrow z^{(l_r)} + \Delta z^{(l_r)}$
13:     $\Delta W_{\text{before}} \leftarrow$ recursively apply this algorithm
14:                 to layers 1 to $l_r - 1$ with target $z^{(l_r)}_{\text{target}}$
15:     $a^{(l_r)} \leftarrow \text{ReLU}(z^{(l_r)}_{\text{target}})$
16:     $\Delta W_{\text{after}} \leftarrow$ recursively apply this algorithm
17:                 to layers $l_r + 1$ to $L$ with input $a^{(l_r)}$ and target $y^*$
18:     $W' \leftarrow W + \Delta W_{\text{before}} + \Delta W_{\text{after}}$
19:     if repair successful and $\|W' - W\|$ is minimal so far then
20:         save $W'$ as best solution
21:     end if
22: end for
23: return best $N'$ with weights $W'$
```

4.7 Advantages

This approach offers several benefits:

- **Scalability:** Layer-wise repair is much faster than full network optimization
- **Minimality:** Uses gradients to identify minimal necessary changes
- **Efficiency:** Combines gradient methods (fast) with optimization (precise)
- **Flexibility:** Can choose which layers to modify based on analysis

5 Results

We evaluated our gradient-based layer-wise repair approach on small neural networks with hand-crafted properties.

5.1 Experimental Setup

We created simple fully-connected networks (2-4 hidden layers, 2-4 neurons per layer with ReLU activations) with well-defined properties. For each counterexample, we applied our repair approach and compared results with optimal solutions computed using Gurobi optimizer.

5.2 Findings

Our approach successfully repaired small networks using gradient-proportional perturbations. The weight updates were comparable to Gurobi's optimal solutions, demonstrating that gradient-based heuristics effectively identify near-optimal repair directions.

5.3 Limitations

Main Limitations:

- The nearest target output y^* might not be unique. Having multiple counterexamples compounds this effect. Layer-wise recursive updates further compound this non-uniqueness.
- Scalability to larger networks (hundreds of layers, thousands of neurons) remains untested
- Recursive repair on deep networks may accumulate approximation errors

5.4 Summary

Our results show that the approach successfully repairs small networks with weight modifications comparable to optimal solutions. The method achieves optimization-quality repairs with gradient-method efficiency, validating that gradient-proportional perturbations can guide minimal repairs.

References

[1] Minimal Modifications of Deep Neural Networks using Verification. https://easychair.org/publications/open/CWhF

[2] Minimal Multi-Layer Modifications of Deep Neural Networks. https://arxiv.org/abs/2110.09929