2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Safe-by-Repair: A Convex Optimization Approach for Repairing Unsafe Two-Level Lattice Neural Network Controllers

Ulices Santa Cruz[1], James Ferlez[1], and Yasser Shoukry[1]

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9993239

***Abstract*— In this paper, we consider the problem of *repairing* a data-trained Rectified Linear Unit (ReLU) Neural Network (NN) controller for a discrete-time, input-affine system. That is, we assume such a NN controller is given, and seek to repair unsafe closed-loop behavior at one known "counterexample" state, without violating closed-loop safety on a separate set of states. Our main result is an algorithm that can systematically and efficiently perform such repair, assuming that the controller has a Two-Level Lattice (TLL) architecture. In particular, we show sufficient conditions for the TLL repair problem can be formulated as two separate, but largely decoupled convex optimization problems: one of essentially local scope and one of essentially global scope. Furthermore, we use our algorithm to repair a TLL controller trained for a four-wheel-car dynamical model.**

I. INTRODUCTION

The proliferation of Neural Networks (NNs) as safety-critical controllers has made obtaining provably correct NN controllers essential. However, most techniques for doing so involve repeatedly training and verifying a NN until safety properties have been achieved. Such methods are computationally expensive (because training and verification are), and their convergence properties can be poor. Moreover, a lack of guarantees means such methods can cycle between safety properties, with each retraining achieving one safety property by undoing another.

An alternative approach obtains a safe NN controller by *repairing* an existing NN controller. Specifically, it is assumed that an already-trained NN controller is available that performs in a *mostly* correct fashion—albeit with some specific, known instances of incorrect behavior. But rather than using retraining techniques, repair entails *systematically* altering the parameters of the original controller *in a limited way*, so as to retain the original safe behavior while simultaneously correcting the unsafe behavior. The objective of repair is to exploit as much as possible the safety that was learned during the training of the original NN parameters, rather than allowing re-training to *unlearn* safe behavior.

In this paper, we exhibit an explicit algorithm that can repair a NN controller for a *discrete-time, input-affine nonlinear* system. Most importantly, however, the repairs affected by our algorithm have *formal safety guarantees* with respect these dynamics. Specifically, our algorithm can repair a NN controller at a particular state in the state space—i.e., make it safe—but without causing a *safety regression* in other states. To this end, our algorithm uses knowledge of the nonlinear dynamics to provably prevent its repair from undoing any initial safety properties of the NN in question. Indeed, our algorithm can be used repeatedly in alternation with a model checker: each counterexample found by the model checker can be repaired, and the resultant controller re-checked to obtain a sequence of repaired controllers, *each safe at strictly more states than the one before*. This distinguishes our approach from CBF-based methods[1], which require a CBF for each extended set.

The cornerstone of our approach is to consider NN controllers of a specific architecture: the recently proposed Two-Level Lattice (TLL) NN architecture [2]. The unique neuronal semantics of the TLL architecture greatly facilitate finding a balance between the considerations inherent in NN repair: i.e., repairing safety at a new state (*local considerations*) vs. preventing safety regressions in other states (*global considerations*). In particular, by assuming a TLL architecture, we can separate the problem of controller repair into two *significantly decoupled* optimization problems, one consisting of essentially only local considerations and one consisting of essentially only global ones. Moreover, these optimization problems are *convex* under the assumption of affine dynamics, which makes our algorithm efficient (especially compared to MIP approaches [1]). Finally, note that considering TLLs is also advantageous in the model-checker context noted above, since recent TLL-specific verifiers (and hence model checkers) perform much better than general NN verifiers [3].

Related Work: Early results on NN repair can be traced back to [4], [5], which focused on patching NN classifiers. Another repair approach used a Satisfiability Modulo Theory formulation with frozen nets in order to perform patching [6]. However, this prior work lacked formal guarantees on the repaired NN. A more formal approach was undertaken in [7], where patching was cast as a formal verification problem for NNs; however, this method focused on a restricted version of the problem where repairs were limited to a single layer. Further work on NN repair has shown promising results, such as gradient guidance to modify the most relevant neurons [8] or [9] which proposes a sound and complete algorithm to synthesize a patch over a problematic region of the NN's input domain. The closest to our work is [1], which uses a Mixed Integer Quadratic Program to satisfy safety specifications, while maintaining performance on a training data set; however, our work differs in the ways noted above.

II. PRELIMINARIES

A. Notation

We will denote the real numbers by $\mathbb{R}$. For an $(n \times m)$ matrix (or vector), A, we will use the notation $[\![A]\!]_{i,j}$ to denote the element in the i^{th} row and j^{th} column of A. Analogously, the notation $[\![A]\!]_{[i,:]}$ will denote the i^{th} row of A, and $[\![A]\!]_{[:,j]}$ will denote the j^{th} column of A; when A is a vector instead, both notations will return a scalar. Let $\mathbf{0}_{n,m}$ be an $(n \times m)$ matrix of zeros, and $\mathbf{I}_n$ be the $(n \times n)$

This work was partially sponsored by the NSF awards #CNS-2002405 and #CNS-2013824 and the C3.AI Digital Transformation Institute.

[1]Ulices Santa Cruz, James Ferlez, and Yasser Shoukry are with the Department of Electrical Engineering and Computer Science, University of California Irvine, Email: {usantacr, jferlez,yshoukry}@uci.edu

identity matrix. $\|\cdot\|$ will refer to the two norm (or the induced two norm for matrices). We will use angle brackets $\langle\cdot\rangle$ to delineate the arguments to a function that *returns a function*. Finally, for $f:\mathbb{R}^n\to\mathbb{R}^m$ and $i\in\{1,\dots,m\}$ define:

$$\pi_i\langle f\rangle : x\mapsto [\![f(x)]\!]_{[i,:]}. \tag{1}$$

B. Dynamical Model

We will consider a discrete-time affine nonlinear system:

$$\Sigma:\{x_{i+1}=f(x_i)+g(x_i)u_i \tag{2}$$

where $x\in\mathbb{R}^n$ is the state, and $u\in\mathbb{R}^m$ is the input. We further assume f and g are smooth functions of x.

Definition 1 (Closed-loop Trajectory)**.** *Let $u:\mathbb{R}^n\to\mathbb{R}^m$. Then a **closed-loop trajectory** of (2) under u, starting from state x_0, will be denoted by the sequence $\{\zeta_i^{x_0}(u)\}_{i=0}^{\infty}$. That is $\zeta_{i+1}^{x_0}(u)=f(\zeta_i^{x_0}(u))+g(\zeta_i^{x_0}(u))\cdot u(\zeta_i^{x_0}(u))$ and $\zeta_0^{x_0}(u)=x_0$.*

Definition 2 (Workspace)**.** *We will assume that trajectories of (2) are confined to a connected, compact workspace, X_{ws} with non-empty interior, of size ext$(X_{\mathsf{ws}})\triangleq\sup_{x\in X_{\mathsf{ws}}}\|x\|$.*

C. Neural Networks

We will exclusively consider Rectified Linear Unit Neural Networks (ReLU NNs). A K-layer ReLU NN is specified by K *layer* functions of two kinds: linear and nonlinear. A layer of either type is defined by a parameter list $\theta\triangleq(W,b)$ where W is a $(\overline{d}\times\underline{d})$ matrix and b is a $(\overline{d}\times 1)$ vector. The *linear* and *nonlinear* layers specified by θ are denoted by $\mathscr{L}_\theta,\mathscr{L}_\theta^\sharp:\mathbb{R}^{\underline{d}}\to\mathbb{R}^{\overline{d}}$, respectively, and they are defined as:

$$\mathscr{L}_\theta: z\mapsto Wz+b \quad\text{and}\quad \mathscr{L}_\theta^\sharp: z\mapsto \max\{\mathscr{L}_\theta(z),0\}, \tag{3}$$

where the max is element-wise. Thus, a K-layer ReLU NN function is formed by composing K layer functions whose parameters $\theta^{|i}, i=1,\dots,K$ have dimensions that satisfy $\underline{d}^{|i}=\overline{d}^{|i-1}: i=2,\dots,K$; we will consistently use the superscript notation $^{|k}$ to identify a parameter with layer k. Furthermore, the linear layers of a NN will be specified by a set $\mathtt{lin}\subseteq\{1,\dots,K\}$. For example, a typical K-layer NN has $\mathtt{lin}=\{K\}$, which together with a list of K layer parameters defines the NN: $\mathscr{NN}=\mathscr{L}_{\theta^{|K}}\circ\mathscr{L}^\sharp_{\theta^{|K-1}}\circ\cdots\circ\mathscr{L}^\sharp_{\theta^{|1}}$. To make the dependence on parameters explicit, we will index a ReLU function $\mathscr{NN}$ by a *list of NN parameters* $\Theta\triangleq(\mathtt{lin},\theta^{|1},\dots,\theta^{|K})$; i.e., $\mathscr{NN}\langle\Theta\rangle:\mathbb{R}^{\underline{d}^{|1}}\to\mathbb{R}^{\overline{d}^{|K}}$.

D. Special NN Operations

Definition 3 (Sequential (Functional) Composition)**.** *Let $\mathscr{NN}\langle\Theta_i\rangle:\mathbb{R}^{\underline{d}_i^{|1}}\to\mathbb{R}^{\overline{d}_i^{|K_i}}$, $i=1,2$ be two NNs with parameter lists $\Theta_i\triangleq(\mathtt{lin}_i,\theta_i^{|1},\dots,\theta_i^{|K_i})$, $i=1,2$ such that $\overline{d}_1^{|K_1}=\underline{d}_2^{|1}$. Then the **sequential (or functional) composition** of $\mathscr{NN}\langle\Theta_1\rangle$ and $\mathscr{NN}\langle\Theta_2\rangle$, i.e. $\mathscr{NN}\langle\Theta_2\rangle\circ\mathscr{NN}\langle\Theta_1\rangle$, is a NN that is represented by the parameter list $\Theta_1\circ\Theta_2\triangleq(\mathtt{lin}_1\cup(\mathtt{lin}_2+K_1),\theta_1^{|1},\dots,\theta_1^{|K_1},\theta_2^{|1},\dots,\theta_2^{|K_2})$, where $\mathtt{lin}_2+K_1$ is an element-wise sum.*

Definition 4. *Let $\mathscr{NN}\langle\Theta_i\rangle:\mathbb{R}^{\underline{d}_i^{|1}}\to\mathbb{R}^{\overline{d}_i^{|K}}$, $i=1,2$ be two K-layer NNs with parameter lists $\Theta_i=(\mathtt{lin},(W_i^{|1},b_i^{|1}),\dots,(W_i^{|K},b_i^{|K}))$, $i=1,2$ such that $\underline{d}_1^{|1}=\underline{d}_2^{|1}$; also note the common set of linear layers, $\mathtt{lin}$. Then the **parallel composition** of $\mathscr{NN}\langle\Theta_1\rangle$ and $\mathscr{NN}\langle\Theta_2\rangle$ is a NN given by:*

$$\Theta_1\parallel\Theta_2\triangleq\left(\mathtt{lin},\left(\begin{bmatrix}W_1^{|1}\\W_2^{|1}\end{bmatrix},\begin{bmatrix}b_1^{|1}\\b_2^{|1}\end{bmatrix}\right),\left(\begin{bmatrix}W_1^{|2}&\mathbf{0}\\\mathbf{0}&W_2^{|2}\end{bmatrix},\begin{bmatrix}b_1^{|2}\\b_2^{|2}\end{bmatrix}\right),\dots,\left(\begin{bmatrix}W_1^{|K}&\mathbf{0}\\\mathbf{0}&W_2^{|K}\end{bmatrix},\begin{bmatrix}b_1^{|K}\\b_2^{|K}\end{bmatrix}\right)\right) \tag{4}$$

where $\mathbf{0}$'s are zero sub-matrices of the appropriate size. That is, $\Theta_1\|\Theta_2$ accepts an input of the same size as (both) Θ_1 and Θ_2, but has as many outputs as Θ_1 and Θ_2 combined.

Definition 5 (n-element min/max NNs)**.** *An n-**element** min **network** is denoted by the parameter list $\Theta_{\min_n}$. $\mathscr{NN}\langle\Theta_{\min_n}\rangle:\mathbb{R}^n\to\mathbb{R}$ such that $\mathscr{NN}\langle\Theta_{\min_n}\rangle(x)$ is the minimum from among the components of x (i.e. minimum according to the usual order relation $<$ on $\mathbb{R}$). An n-**element** max **network** is denoted by $\Theta_{\max_n}$, and functions analogously. These networks are described in [2].*

E. Two-Level-Lattice (TLL) Neural Networks

In this paper, we will consider only Two-Level Lattice (TLL) ReLU NNs [2]. We define a TLL NN as follows.

Definition 6 (TLL NN [2, Theorem 2])**.** *A NN that maps $\mathbb{R}^n\to\mathbb{R}$ is said to be **TLL NN of size** (N,M) if its parameter list $\Xi_{N,M}$ can be written as follows:*

$$\Xi_{N,M}\triangleq\Theta_{\max_M}\circ\left((\Theta_{\min_N}\circ\Theta_{S_1})\parallel\dots\parallel(\Theta_{\min_N}\circ\Theta_{S_M})\right)\circ\Theta_\ell \tag{5}$$

where

- $\Theta_\ell\triangleq(\{1\},\theta_\ell)$ *for* $\theta_\ell\triangleq(W_\ell,b_\ell)$*;*
- *each* Θ_{S_j} *has the form* $\Theta_{S_j}=\left(\{1\},\left(S_j,\mathbf{0}_{N,1}\right)\right)$ *where*
  - $S_j=\left[\,[\![\mathbf{I}_N]\!]_{[\iota_1,:]}^T\ \cdots\ [\![\mathbf{I}_N]\!]_{[\iota_N,:]}^T\,\right]^T$ *for a length-N sequence $\{\iota_k\}$ where $\iota_k\in\{1,\dots,N\}$.*

*Θ_ℓ (and its parameters) constitute the **linear layer** of $\Xi_{N,M}$; the S_j are the **selector matrices** of $\Xi_{N,M}$.*

*A **multi-output TLL NN** with range space $\mathbb{R}^m$ is defined as m equally sized scalar TLL NNs. We denote such a TLL by $\Xi_{N,M}^{(m)}$, with each output given by $\Xi_{N,M}^{\kappa}$, $\kappa=1,\dots,m$.*

Definition 7 (Selector Set)**.** *Let S_j be a selector matrix of a TLL $\Xi_{N,M}$. Then the corresponding **selector set** of S_j is:*

$$s_j\triangleq\{k\in\{1,\dots,N\}|\exists\iota\in\{1,\dots,N\}.[\![S_j]\!]_{[\iota,k]}=1\} \tag{6}$$

Remark 1. *For an input $x\in\mathbb{R}^{\underline{d}^{|1}}$, each composition $\Theta_{\min_N}\circ\Theta_{S_j}, j=1,\dots,M$ in (5) computes $\min_{i\in s_j}\pi_i\langle\mathscr{L}_{\Theta_\ell}\rangle(x)$.*

III. Problem Formulation

The main problem we consider in this paper is one of TLL NN *repair*. In brief, we take as a starting point a TLL NN controller that is "mostly" correct in the sense that is provably safe for a particular, fixed set of states. Then we further suppose that some additional, *unsafe* behavior of said controller is explicitly observed at some other state. The repair problem, then, is to "repair" the given TLL controller so that this additional unsafe behavior is made safe, while simultaneously preserving the original safety guarantees associated with the network.

The basis for the problem in this paper is thus a TLL NN controller that has been designed (or trained) to control (2) in a *safe way*. In particular, we use the following definition to fix our notion of "unsafe" behavior for (2).

Definition 8 (Unsafe Operation of (2))**.** *Let G_u be an $(\mathsf{K}_u\times n)$ matrix, and let h_u be an $(\mathsf{K}_u\times 1)$ vector, which together define a set of **unsafe states** $X_{\mathsf{unsafe}}\triangleq\{x\in\mathbb{R}^n|G_ux\geq h_u\}$.*

Then, we mean that a TLL NN controller is safe with respect to (2) and X_{unsafe} in the following sense.

Definition 9 (Safe TLL NN Controller)**.** *Let $X_{safe} \subset \mathbb{R}^n$ be a set of states such that $X_{safe} \cap X_{unsafe} = \emptyset$. Then a TLL NN controller $\mathfrak{u} \triangleq \mathscr{NN}\langle \Xi^{(m)}_{N,M} \rangle : \mathbb{R}^n \to \mathbb{R}^m$ is **safe** for (2) **on horizon** T (with respect to X_{safe} and X_{unsafe}) if:*

$$\forall x_0 \in X_{safe}, i \in \{1, ..., T\}. \left(\zeta_i^{x_0}(\mathscr{NN}\langle \Xi^{(m)}_{N,M} \rangle) \notin X_{unsafe}\right). \quad (7)$$

That is $\mathscr{NN}\langle \Xi^{(m)}_{N,M} \rangle$ is safe (w.r.t. X_{safe}) if all of its length-T trajectories starting in X_{safe} avoid the unsafe states X_{unsafe}.

The design of safe controllers in the sense of Definition 9 has been considered in a number of contexts; see e.g. [10]. Often a NN is trained on using data collected from an expert, and then verified using one the available NN verifiers [10].

However, we further suppose that a given TLL which is safe in the sense of Definition 9 nevertheless has some *unsafe* behavior for states outside of X_{safe}. In particular, we suppose that we have a *counterexample* to safe operation of (2).

Definition 10 (Counterexample to Safe Operation of (2))**.** *Let $X_{safe} \subset \mathbb{R}^n$, and let $\mathfrak{u} \triangleq \mathscr{NN}\langle \Xi^{(m)}_{N,M} \rangle$ be a TLL controller that is safe for (2) on horizon T w.r.t X_{safe} and X_{unsafe}. A **counter example to the safe operation of** (2) is a state $x_{c.e.} \notin X_{safe}$ such that*

$$f(x_{c.e.}) + g(x_{c.e.}) \cdot \mathfrak{u}(x_{c.e.}) = \zeta_1^{x_{c.e.}}(\mathfrak{u}) \in X_{unsafe}. \quad (8)$$

i.e., starting (2) in $x_{c.e.}$ leads to an unsafe state at $t = 1$.

We can now state the main problem of this paper.

Problem 1. *Let dynamics (2) be given, and assume its trajectories are confined to compact subset of states, X_{ws} (see Definition 2). Also, let $X_{unsafe} \subset X_{ws}$ be a specified set of unsafe states for (2), as in Definition 8. Furthermore, let $\mathfrak{u} = \mathscr{NN}\langle \Xi^{(m)}_{N,M} \rangle$ be a TLL NN controller for (2) that is safe on horizon T with respect to a set of states $X_{safe} \subset X_{ws}$ (see Definition 9), and let $x_{c.e.}$ be a counterexample to safety in the sense of Definition 10.*

*Then the **TLL repair problem** is to obtain a new TLL controller $\bar{\mathfrak{u}} = \mathscr{NN}\langle \bar{\Xi}^{(m)}_{N,M} \rangle$ with the following properties:*

- *(i) $\bar{\mathfrak{u}}$ is also safe on horizon T with respect to X_{safe};*
- *(ii) the trajectory $\zeta_1^{x_{c.e.}}(\bar{\mathfrak{u}})$ is safe – i.e. the counterexample $x_{c.e.}$ is "repaired";*
- *(iii) $\bar{\Xi}^{(m)}_{N,M}$ and $\Xi^{(m)}_{N,M}$ share a common architecture (as implied by their identical architectural parameters); and*
- *(iv) the selector matrices of $\bar{\Xi}^{(m)}_{N,M}$ and $\Xi^{(m)}_{N,M}$ are identical – i.e. $\bar{S}_k = S_k$ for $k = 1, \ldots, M$; and*
- *(v) $\|\bar{W}_\ell - W_\ell\| + \|\bar{b}_\ell - b_\ell\|$ is minimized.*

In particular, *iii)*, *iv)* and *v)* justify the designation of this problem as one of "repair". That is, to fix the counterexample while keeping the network as close as possible to the original network under consideration.

IV. Framework

The TLL NN repair problem described in Problem 1 is challenging because it has two main objectives, which are at odds with each other. In particular, repairing a counterexample requires altering the NN's response in a *local* region of the state space, but changing even a few neurons generally affects the *global* response of the NN – which could undo the initial safety guarantee supplied with the network. This tension is especially relevant for general deep NNs, and repairs realized on neurons in their latter layers. It is for this reason that we posed Problem 1 in terms of TLL NNs: our approach will be to use the unique semantics of TLL NNs to balance the trade-offs between **local NN alteration to repair the defective controller** and **global NN alteration to ensure that the repaired controller activates at the counterexample**. Moreover, locally repairing the defective controller at $x_{c.e.}$ entails a further trade off between two competing objectives of its own: actually repairing the counterexample – Problem 1*(ii)* – without causing a violation of the original safety guarantee for X_{safe} – i.e. Problem 1*(i)*. Likewise, global alteration of the TLL to ensure correct activation of our repairs will entail its own trade-off: the alterations necessary to achieve the correct activation cannot sacrifice the safety guarantee for X_{safe} – i.e. Problem 1*(i)*.

A. Local TLL Repair

We first consider in isolation the problem of repairing the TLL controller in the vicinity of the counterexample $x_{c.e.}$, but under the assumption that the altered controller will remain the active there. The problem of actually guaranteeing that this is the case will be considered in the subsequent section. Thus, our repair **establishes constraints** on the alterations of those parameters in the TLL controller associated with the affine controller instantiated at and around the state $x_{c.e.}$.

Definition 11 (Local Linear Function)**.** *Let $\Psi : \mathbb{R}^n \to \mathbb{R}$ be continuous piecewise affine (CPWA). Then a **local linear function of** Ψ is an affine function $l : \mathbb{R}^n \to \mathbb{R}$ if there exists an open set $\mathfrak{O}$ such that $l(x) = \Psi(x)$ for all $x \in \mathfrak{O}$.*

The unique semantics of TLL NNs makes them especially well suited to this local repair task because in a TLL NN, its local linear functions appear directly as neuronal parameters. In particular, the local linear functions of a TLL NN are described *directly* by parameters in its linear layer; i.e. $\Theta_\ell = (W_\ell, b_\ell)$ for a scalar TLL NN (see Definition 6). This follows as a corollary the following proposition in [11]:

Proposition 1 ([11, Proposition 3])**.** *Let $\Xi_{N,M}$ be a TLL with $\theta_\ell = (W_\ell, b_\ell)$, and define:*

$$\ell_i \triangleq \pi_i\langle \mathscr{L}_{\theta_\ell} \rangle = x \mapsto w_{\ell,i}x + b_{\ell,i} \quad i = 1, \ldots, N \quad (9)$$

where $w_{\ell,i} \triangleq [\![W_\ell]\!]_{[i,:]}$ and $b_{\ell,i} \triangleq [\![b_\ell]\!]_i$. Then each local linear function of $\mathscr{NN}\langle \Xi_{N,M} \rangle$ is equal to ℓ_i for some $i \in \{1, \ldots, N\}$.

Definition 12 (Local Linear Functions of a TLL)**.** *Because of Proposition 1, the ℓ_i defined in (9) are referred to as as the **local linear functions** of the TLL $\Xi_{N,M}$.*

Corollary 1. *Let $\Xi^{(m)}_{N,M}$ be a TLL on $\mathbb{R}^n$, and let $x_{c.e.} \in \mathbb{R}^n$. Then there exist integers $act_k \in \{1, \ldots, N\}$ for $\kappa = 1, \ldots, m$ and a connected open set $R_a \subset \mathbb{R}^n$ such that*

- *$x_{c.e.}$ is in the closure of R_a; and*
- *$\pi_\kappa\langle \mathscr{NN}\langle \Xi^{(m)}_{N,M} \rangle \rangle = \mathscr{NN}\langle \Xi^{\kappa}_{N,M} \rangle = \ell^{\kappa}_{act_k}$ on the set R_a.*

Corollary 1 is actually a strong statement: it indicates that in a TLL, each local linear function is described directly by *its own* linear-function-layer parameters and those parameters describe *only* that local linear function.

As a consequence of Corollary 1, "repairing" the problematic local linear function of the TLL controller in Problem 1 involves the following steps:

1) identify which of the local linear functions is realized by the TLL controller at $x_{\text{c.e.}}$ – i.e. identifying the indices of the active local linear function at $x_{\text{c.e.}}$ viz. indices $\text{act}_\kappa \in \{1,\dots,N\}$ for each output κ as in Corollary 1;
2) *establish constraints* on the parameters of that local linear function so as to ensure repair of the counterexample; i.e. altering the elements of the rows $w^\kappa_{\ell,\text{act}_\kappa} = [\![W^\kappa_\ell]\!]_{[\text{act}_\kappa,:]}$ and $b^\kappa_{\ell,\text{act}_\kappa} = [\![b^\kappa_\ell]\!]_{\text{act}_\kappa}$ for each output κ such that the resulting linear controller repairs the counterexample as in Problem 1*(ii)*; and
3) *establish constraints* to ensure the repaired parameters do not induce a violation of the safety constraint for the guaranteed set of safe states, X_{safe}, as in Problem 1*(i)*.

We consider these three steps in sequence as follows.

1) Identifying the Active Controller at $x_{\text{c.e.}}$: From Corollary 1, *all* of the possible linear controllers that a TLL controller realizes are exposed directly in the parameters of its linear layer matrices, Θ^κ_ℓ. Crucially for the repair problem, once the active controller at $x_{\text{c.e.}}$ has been identified, the TLL parameters responsible for that controller are immediately evident. This is the starting point for our repair process. Since a TLL consists of two levels of lattice operations, it is straightforward to identify which of these affine functions is in fact active at $x_{\text{c.e.}}$; for a given output, κ, this is can be done by evaluating $W^\kappa_\ell x_{\text{c.e.}} + b^\kappa_\ell$ and comparing the components thereof according to the selector sets associated with the TLL controller. That is the index of the active controller for output κ, denoted by act_κ, is determined by:

$$\text{act}_\kappa \triangleq \underset{i\in\{\mu^\kappa_j|j=1,\dots,M\}}{\arg\max} \ell^\kappa_j(x_{\text{c.e.}}) \text{ for } \mu^\kappa_j \triangleq \arg\min_{i\in S^\kappa_j} \ell^\kappa_i. \quad (10)$$

These expressions mirror the computations that define a TLL, as described in Definition 6; the only difference is that $\max$ and $\min$ are replaced by $\arg\max$ and $\arg\min$, respectively.

2) Repairing the Affine Controller at $x_{\text{c.e.}}$: From Corollary 1, the parameters of the network that result in a problematic controller at $x_{\text{c.e.}}$ are readily apparent. Moreover, these parameters are in the linear layer of the original TLL, they are alterable under the requirement in Problem 1. Thus, local repair entails simply correcting the elements of the matrices $w^\kappa_{\ell,\text{act}_\kappa} = [\![W^k_\ell]\!]_{[\text{act}_\kappa,:]}$ and $b^\kappa_{\ell,\text{act}_\kappa} = [\![b^\kappa_\ell]\!]_{\text{act}_k}$. It is thus clear that a "repaired" controller should satisfy

$$f(x_{\text{c.e.}}) + g(x_{\text{c.e.}}) \left[\, \ell^1_{\text{act}_1}(x_{\text{c.e.}}) \;\dots\; \ell^m_{\text{act}_m}(x_{\text{c.e.}}) \,\right]^{\mathrm{T}} \notin X_{\text{unsafe}}. \quad (11)$$

Then (11) represents a *linear constraint* in the local controller to be repaired, and this constraint imposes the repair property in Problem 1*(ii)*.

3) Preserving the Initial Safety Condition with the Repaired Controller: The fact that we *altered* the controller parameters thus means that trajectories emanating from X_{safe} may be affected in turn by our repair efforts: that is the repairs made to address Problem 1*(ii)* may simultaneously alter the TLL in a way that **undoes** the requirement in Problem 1*(i)* – i.e. the initial safety guarantee on X_{safe} and $\mathscr{M}\langle\Xi^{(m)}_{N,M}\rangle$. Thus, local repair of the faulty controller must account for this safety property, too.

We accomplish this by bounding the reach set of (2) for initial conditions in X_{safe}, and for this we employ the usual strategy of bounding the relevant Lipschitz constants. Since the TLL controller is a CPWA controller, these bounds will incorporate the TLL's local linear functions via $\|w^\kappa_{\ell,i}\|$ and $\|b^\kappa_{\ell,i}\|$ for $i \in \{1,\dots,N\}, \kappa \in \{1,\dots,m\}$ (Definition 12).

Proposition 2. *Consider system dynamics (2), and suppose that the state x is confined to known compact workspace, X_{ws} (see Definition 2). Also, let T be the integer time horizon from Definition 9. Finally, assume that a closed-loop CPWA $\Psi : \mathbb{R}^n \to \mathbb{R}^m$ is applied to (2), and that Ψ has local linear functions $\mathcal{L}_\Psi = \{x \mapsto w_k x + b_k | k = 1,\dots,N\}$.*

Moreover, define the function β as

$$\beta(\|w\|,\|b\|) \triangleq \sup_{x_0\in X_{safe}} \Big(\|f(x_0) - x_0\| + \|g(x_0)\|\cdot\|w\|\cdot ext(X_{safe}) + \|g(x_0)\|\cdot\|b\| \Big) \quad (12)$$

and in turn define

$$\beta_{max}(\Psi) \triangleq \beta\big(\max_{w\in\{w_k|k=1,\dots,N\}} \|w\|, \max_{b\in\{b_k|k=1,\dots,N\}} \|b\| \big). \quad (13)$$

Finally, define the function L as in (14), and in turn define

$$L_{max}(\Psi) \triangleq L\big(\max_{w\in\{w_k|k=1,\dots,N\}} \|w\|, \max_{b\in\{b_k|k=1,\dots,N\}} \|b\| \big). \quad (15)$$

Then for all $x_0 \in X_{safe}$, $i \in \{1,\dots,T\}$, we have:

$$\|\zeta^{x_0}_T(\Psi) - x_0\| \leq \beta_{max}(\Psi)\cdot\sum_{k=0}^{T} L_{max}(\Psi)^k. \quad (16)$$

Proposition 2 bounds the size of the reach set for (2) in terms of an arbitrary CPWA controller, Ψ, when the system is started from X_{safe}. This proposition is naturally applied to find bounds for safety with respect to X_{unsafe} as follows.

Proposition 3. *Let T, X_{ws}, Ψ and $\mathcal{L}_\Psi$ be as in Proposition 2, and let β_{max} and L_{max} be two constants s.t. for all $\delta x \in \mathbb{R}^n$*

$$\|\delta x\| \leq \beta_{max}\sum_{k=0}^{T} L_{max}{}^k \Longrightarrow \forall x_0 \in X_{safe}.\big(x_0 + \delta x \notin X_{unsafe}\big) \quad (17)$$

If $\beta_{max}(\Psi) \leq \beta_{max}$ and $L_{max}(\Psi) \leq L_{max}$, then trajectories of (2) under controller Ψ are safe in the sense that

$$\forall x_0 \in X_{safe} \forall i \in \{1,\dots,T\} \,.\, \zeta^{x_0}_t(\Psi) \notin X_{unsafe}. \quad (18)$$

In particular, Proposition 3 states: if we find constants $\beta_{\max}$ and $L_{\max}$ that satisfy (17), then we have a way to bound the parameters of any CPWA controller (via β and L) so that that controller is safe in closed loop. Indeed, this is a sufficient condition to preserve the safety property required in Problem 1*(i)*, so it can be imposed on the repaired TLL controller.

Formally, this entails particularizing Proposition 2 and 3 to the TLL controllers associated with the repair problem.

Corollary 2. *Again consider system (2) confined to workspace X_{ws} as before. Also, let β_{max} and L_{max} be such that they satisfy the assumptions of Proposition 3, viz. (17).*

Now, let $\Xi^{(m)}_{N,M}$ be the TLL controller given in Problem 1, and let $w^\kappa_{\ell,i}$ and $b^\kappa_{\ell,i}$, characterize its local linear functions for $i = 1,\dots,N$ and outputs $\kappa = 1,\dots,m$ (see Definition 12). For this controller, define:

$$\Omega_W \triangleq \max_{w\in\cup^m_{\kappa=1}\{w^\kappa_{\ell,i}|i=1,\dots,N\}} \|w\| \quad (19)$$

$$\Omega_b \triangleq \max_{b\in\cup^m_{\kappa=1}\{b^\kappa_{\ell,i}|i=1,\dots,N\}} \|b\| \quad (20)$$

so that $\beta_{max}(\Xi^{(m)}_{N,M}) = \beta(\Omega_W,\Omega_b)$ and $L_{max}(\Xi^{(m)}_{N,M}) = L(\Omega_W,\Omega_b)$. Finally, let indices $\{act_\kappa\}^m_{\kappa=1}$ specify the active

local linear functions of $\Xi^{(m)}_{N,M}$ that are to be repaired, as described in Subsection IV-A.1 and IV-A.2. Let $\bar{w}^{\kappa}_{\ell,act_\kappa}$ and $\bar{b}^{\kappa}_{\ell,act_\kappa}$ be repaired values $w^{\kappa}_{\ell,act_\kappa}$ and $b^{\kappa}_{\ell,act_\kappa}$, respectively

If the following four conditions are satisfied

$$\beta(\|\bar{w}^{\kappa}_{act_\kappa}\|, \|\bar{b}^{\kappa}_{act_\kappa}\|) \leq \beta_{max} \tag{21}$$

$$\beta_{max}(\Xi^{(m)}_{N,M}) \leq \beta_{max} \tag{22}$$

$$L(\|\bar{w}^{\kappa}_{act_\kappa}\|, \|\bar{b}^{\kappa}_{act_\kappa}\|) \leq L_{max} \tag{23}$$

$$L_{max}(\Xi^{(m)}_{N,M}) \leq L_{max} \tag{24}$$

then the following hold for all $x_0 \in X_{safe}$:

$$\|\zeta^{x_0}_T(\bar{\Xi}^{(m)}_{N,M}) - x_0\| \leq \beta_{max} \cdot \sum_{k=0}^{T} {L_{max}}^k \tag{25}$$

and hence

$$\forall i \in \{1, \ldots, T\} \,.\, \zeta^{x_0}_i(\bar{\Xi}^{(m)}_{N,M}) \notin X_{unsafe}. \tag{26}$$

The conclusion (25) of Corollary 2 should be interpreted as follows: the bound on the reach set of the repaired controller, $\bar{\Xi}^{(m)}_{N,M}$, is no worse than the bound on the reach set of the original TLL controller given in Problem 1. Hence, by the assumptions borrowed from Proposition 3, conclusion (26) of Corollary 2 indicates that **the repaired controller $\bar{\Xi}^{(m)}_{N,M}$ remains safe in the sense of** Problem 1*(i)* – i.e. closed-loop trajectories emanating from X_{safe} remain safe on horizon T.

In the sequel, (21) and (23) will play the crucial role of ensuring that the repaired controller respects Problem 1*(i)*.

B. Global TLL Alteration for Repaired Controller Activation

Analogous to the local alteration consider before, we need to **devise global constraints sufficient to enforce the activation of the repaired controller at** $x_{\text{c.e.}}$. Thus, ensuring that a particular, indexed local linear function is active at the output of a TLL entails ensuring that that function

(*a*) appears at the output of one of the $\min$ networks; and
(*b*) appears at the output of the $\max$ network, by exceeding the outputs of all the *other* $\min$ networks.

Notably, this sequence also suggests a means for ensuring that the repaired controller remains active at the counter example. Formally, we have the following proposition.

Proposition 4. *Let $\Xi^{(m)}_{N,M}$ be a TLL NN with local linear functions characterized by $w^{\kappa}_{\ell,i}$ and $b^{\kappa}_{\ell,i}$, and let $x_{\text{c.e.}} \in \mathbb{R}^n$.*

Then the index $act_\kappa \in \{1, \ldots, N\}$ denotes the local linear function that is active at $x_{\text{c.e.}}$ for output κ (see Corollary 1), if and only if there exists a $sel_\kappa \in \{1, \ldots, M\}$ such that

(i) for all $i \in S^{\kappa}_{sel_\kappa}$ and any $x \in R_a$, $\ell^{\kappa}_{act_\kappa}(x) \leq \ell^{\kappa}_i(x)$ i.e. the active local linear function "survives" the $\min$ network associated with selector set $S^{\kappa}_{sel_\kappa}$; and

(ii) for all $j \in \{1, \ldots, M\} \backslash \{sel_\kappa\}$ there exists an index $\iota^{\kappa}_j \in \{1, \ldots, N\}$ s.t. for all $x \in R_a$, $\ell^{\kappa}_{\iota^{\kappa}_j}(x) \leq \ell^{\kappa}_{act_\kappa}(x)$ i.e. the active local linear function "survives" the $\max$ network of output κ by exceeding the output of all of the other $\min$ networks.

These alterations clearly must be made to local linear functions which are not active at the counterexample, hence terminology "global alteration".

Finally, note that altering these un-repaired local linear functions – i.e. those not indexed by act_κ – may create the same issue described in Section IV-A.3. Thus, for any of these global alterations additional safety constraints like (21) and (23) must be imposed on the altered parameters.

V. Main Algorithm

Problem 1 permits the alteration of linear-layer parameters in the original TLL controller to perform repair. Thus, the core of our algorithm is to employ a convex solver to find the minimally altered TLL parameters that also satisfy the local and global constraints we have outlined for successful repair with respect to the other aspects of Problem 1. The fact that the local repair constraints are prerequisite to the global activation constraints means that we will employ a convex solver on two optimization problems *in sequence*: first, to determine the feasibility of local repair and effectuate that repair in a minimal way; and then subsequently to determine the feasibility of activating said repaired controller as required and effectuating that activation in a minimal way.

Remark 2. *We will use $\bar{W}^{\kappa}_{\ell}$ and $\bar{b}^{\kappa}_{\ell}$ to characterize the linear layer of the repaired TLL, with $\bar{w}^{\kappa}_{\ell,i} \triangleq [\![\bar{W}^{\kappa}_{\ell}]\!]_{[i,:]}$, $\bar{b}^{\kappa}_{\ell,i} \triangleq [\![\bar{b}^{\kappa}_{\ell}]\!]_i$ and $\bar{\ell}^{\kappa}_i$ suitably defined; c.f. Definition 12 and Corollary 2.*

A. Optimization Problem for Local Alteration (Repair)

Local alteration for repair starts by identifying the active controller at the counterexample, as denoted by the index act_κ for each output of the controller, κ. From this knowledge, an explicit constraint sufficient to repair the local controller at $x_{\text{c.e.}}$ is specified by the dynamics: see (11).

Our formulation of a safety constraint for the locally repaired controller requires additional input, though. In particular, we need to identify constants $\beta_{\max}$ and $L_{\max}$ such that the non-local controllers satisfy (22) and (24). Then Corollary 2 implies that (21) and (23) are constraints that ensure the *locally* repaired controller satisfies Problem 1*(i)*. For this we take the naive approach of setting $\beta_{\max} = \beta(\Xi^{(m)}_{N,M})$, and then solving for the smallest $L_{\max}$ that ensures safety for that particular $\beta_{\max}$. In particular, we set

$$L_{\max} = \inf\{L' > 0 \mid \beta_{\max} \cdot \sum_{k=0}^{T} L'^k = \inf_{\substack{x_s \in X_{\text{safe}} \\ x_u \in X_{\text{unsafe}}}} \|x_s - x_u\|\}. \tag{27}$$

Thus, we formulate the local repair optimization problem:

$$\begin{aligned}
\texttt{Local}: \min_{\bar{w}^{\kappa}_{\text{act}_\kappa}, \bar{b}^{\kappa}_{\text{act}_\kappa}} \; & \sum_{\kappa=1}^{m} \|w^{\kappa}_{\ell,\text{act}_\kappa} - \bar{w}^{\kappa}_{\ell,\text{act}_\kappa}\| + \|b^{\kappa}_{\ell,\text{act}_\kappa} - \bar{b}^{\kappa}_{\ell,\text{act}_\kappa}\| \\
\text{s.t.} \; & f(x_{\text{c.e.}}) + g(x_{\text{c.e.}}) \left[\bar{\ell}^1_{\text{act}_1}(x_{\text{c.e.}}) \; \ldots \; \bar{\ell}^m_{\text{act}_m}(x_{\text{c.e.}}) \right]^{\mathrm{T}} \notin X_{\text{unsafe}} \\
& \forall \kappa \in \{1, \ldots, m\} \;.\; L(\|\bar{w}^{\kappa}_{\ell,\text{act}_\kappa}\|, \|\bar{b}^{\kappa}_{\ell,\text{act}_\kappa}\|) \leq L_{\max} \\
& \forall \kappa \in \{1, \ldots, m\} \;.\; \beta(\|\bar{w}^{\kappa}_{\ell,\text{act}_\kappa}\|, \|\bar{b}^{\kappa}_{\ell,\text{act}_\kappa}\|) \leq \beta_{\max} \\
& \forall \kappa \in \{1, \ldots, m\} \;.\; L(\bar{\Xi}^{\kappa}_{N,M}) \leq L_{\max}
\end{aligned}$$

$$L(\|w\|, \|b\|) \triangleq L_f + L_g \cdot \sup_{x_0 \in X_{\text{ws}}} \|w\| \cdot \|x_0\| + \sup_{x_0 \in X_{\text{ws}}} \|w\| \cdot \|g(x_0)\| + L_g \cdot \|b\| \tag{14}$$

B. Optimization Problem for Global Alteration (Activation)

If **Local** is feasible, the local controller at $x_{\text{c.e.}}$ can be repaired, and the global activation of said controller can be considered. Since we start with a local linear function we desire to be active at $x_{\text{c.e.}}$, we retain the definition of act_κ from the **Local** initialization. Moreover, Problem 1 preserves the selector matrices of the original TLL controller, thus we define the selector indices, sel_κ, in terms of the activation pattern of the *original*, defective local controller.

Thus, in order to formulate an optimization problem for global alteration, we need to define constraints compatible with Proposition 4 based on the activation/selector indices described above. Part *(i)* of the conditions in Proposition 4 is unambiguous at this point: it says that the desired active local linear function, act_κ, must have the minimum output from among those functions selected by selector set s_{sel_κ}. Part *(ii)* of the conditions in Proposition 4 is ambiguous however: we only need to specify *one* local linear function from each of the *other* min groups to be "forced" lower than the desired active local linear function. In the face of this ambiguity, we select these functions using indices $\iota_j^\kappa : j \in \{1,\dots,M\}\backslash\{\text{act}_\kappa\}$ that are defined as follows:

$$\iota_j^\kappa \triangleq \arg\min_{i \in s_j^\kappa} \ell_i^\kappa(x_{\text{c.e.}}). \tag{28}$$

That is, we form our global alteration constraints out of the non-active controllers that have the *lowest outputs among their respective min groups*.

Thus, we formulate the global alteration optimization as:

$$\textbf{Global}: \min_{\bar{W}_\ell^\kappa, \bar{b}_\ell^\kappa} \sum_{\kappa=1}^{m} \|W_\ell^\kappa - \bar{W}_\ell^\kappa\| + \|b_\ell^\kappa - \bar{b}_\ell^\kappa\|$$

$$\begin{aligned}
\text{s.t.}\quad & \forall \kappa \in \{1,\dots,m\} && . \ w^\kappa_{\ell,\text{act}_\kappa} = \bar{w}^\kappa_{\ell,\text{act}_\kappa} \\
& \forall \kappa \in \{1,\dots,m\} && . \ b^\kappa_{\ell,\text{act}_\kappa} = \bar{b}^\kappa_{\ell,\text{act}_\kappa} \\
& \forall \kappa \in \{1,\dots,m\} \ \forall i \in s_{\text{sel}_\kappa} && . \ \bar{\ell}_{\text{act}_\kappa}(x_{\text{c.e.}}) \le \ell_i(x_{\text{c.e.}}) \\
& \forall \kappa \in \{1,\dots,m\} \forall j \in \{1,\dots,M\}\backslash\{\text{sel}_\kappa\} && . \ \ell_{\iota_j^\kappa}(x_{\text{c.e.}}) \le \bar{\ell}_{\text{act}_\kappa}(x_{\text{c.e.}}) \\
& \forall \kappa \in \{1,\dots,m\} \ \forall i \in \{1,\dots,N\} && . \ L(\|\bar{w}^\kappa_{\ell,i}\|, \|\bar{b}^\kappa_{\ell,i}\|) \le L_{\max} \\
& \forall \kappa \in \{1,\dots,m\} \ \forall i \in \{1,\dots,N\} && . \ \beta(\|\bar{w}^\kappa_{\ell,i}\|, \|\bar{b}^\kappa_{\ell,i}\|) \le \beta_{\max} \\
& \forall \kappa \in \{1,\dots,m\} && . \ L(\bar{\Xi}^\kappa_{N,M}) \le L_{\max}
\end{aligned}$$

where $\bar{w}^\kappa_{\text{act}_\kappa}$ and $\bar{b}^\kappa_{\text{act}_\kappa}$ are the repaired local controller parameters from **Local**.

VI. Numerical Examples

We tested our framework on a four-wheel car with the model:

$$x(t+1) = \begin{bmatrix} x_1(t) + V\cos(x_3(t))\cdot t_s \\ x_2(t) + V\sin(x_3(t))\cdot t_s \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ t_s \end{bmatrix} v(t) \tag{29}$$

where the state is $x(t) = [p_x(t)\ p_y(t)\ \Psi(t)]^T$ with position $(p_x\ p_y)$ and angle Ψ; the control input v is the angle rate. The parameters are the translational speed, V (meters/sec); and the sampling period, t_s (sec). For our experiments, $V = 0.3$ m/s and $t_s = 0.01$ sec, the workspace $X_{\text{ws}} = [-3,3]\times[-4,4]\times[-\pi,\pi]$; safe states $X_{\text{safe}} = [-0.25, 0.25]\times[-0.75,-0.25]\times[-\frac{\pi}{8},\frac{\pi}{8}]$, verified using NNV [10] over 100 iterations; and unsafe set X_{unsafe} specified by $[0\ 1\ 0]\cdot x > 3$.

All experiments were executed on an Intel Core i5 with 8 GB of memory. First, we collected 1850 state-action data pairs from a PI Controller to steer the car and avoid X_{unsafe}, to train a TLL NN of size $N = 50, M = 10$ on a corrupted version of this data-set: by manually changing the control action on 25 data points close to X_{unsafe}. Using this TLL NN controller for different x_0 we identified $x_{\text{c.e.}} = [0\ 2.999\ 0.2]$ as a valid counterexample for safety after two time steps. Finally, to repair this *faulty* NN, we computed the safety bounds $\beta_{\max} = 0.0865$ and $L_{\max} = 1.42$ for a horizon of $T = 7$. Then, from $x_{\text{c.e.}}$ we obtained the controller $K = [K_w\ K_b]$ where $K_w = [-0.14,\ -0.54,\ -0.42]$ and $K_b = [2.22]$.

Next, we ran our algorithm to repair the counterexample using CVX (convex solver). The result of the first optimization problem, **Local**, was the linear controller: $\bar{K}_w = [-0.002\ \ -0.04\ \ -0.01]$ and $\bar{K}_b = [-9.78]$; this optimization required a total execution time of 1.89 sec. The second optimization problem, **Global**, successfully activated the repaired controller with optimal cost 8.97; this optimization required a total execution time of 6.53 sec. The repaired TLL had $||\bar{W}|| = 11.02$ and $||\bar{b}|| = 5.687$ vs. the original with $||W|| = 6.54$ and $||b|| = 5.6876$.

Finally, we simulated the car using the repaired TLL controller for 50 steps. Fig. 1 shows the state trajectories of both the original, *faulty* controller and the repaired controller. Note: the repaired controller met the safety specifications.

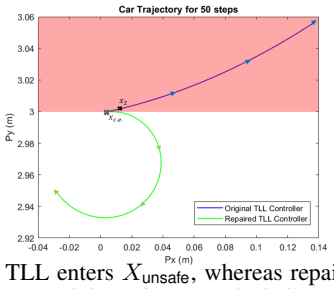

Fig. 1: Original TLL enters X_{unsafe}, whereas repaired TLL is safe. Red area is X_{unsafe}; Red Cross is $x_{\text{c.e.}}$; Black Cross is $\zeta_2^{x_{\text{c.e.}}}(\bar{\Xi}_{N,M})$.

References

[1] K. Majd, S. Zhou, H. Ben Amor, G. Fainekos, and S. Sankaranarayanan, "Local repair of neural networks using optimization." https://arxiv.org/abs/2109.14041, 2021.

[2] J. Ferlez and Y. Shoukry, "AReN: Assured ReLU NN Architecture for Model Predictive Control of LTI Systems," in *Hybrid Systems: Computation and Control 2020 (HSCC'20)*, ACM, 2020.

[3] J. Ferlez, H. Khedr, and Y. Shoukry, "Fast BATLLNN: Fast Box Analysis of Two-Level Lattice Neural Networks," in *Hybrid Systems: Computation and Control 2022 (HSCC'22)*, ACM, 2022.

[4] S. Kauschke and J. Fürnkranz, "Batchwise patching of classifiers," *The 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[5] S. Kauschke and D. H. Lehmann, "Towards neural network patching: Evaluating engagement-layers and patch-architectures," 2018. http://arxiv.org/abs/1812.03468.

[6] M. Sotoudeh and A. V. Thakur, "Correcting deep neural networks with small, generalizing patches," in *NeurIPS 2019 Workshop on Safety and Robustness in Decision Making*, 2019.

[7] B. Goldberger, Y. Adi, J. Keshet, and G. Katz, "Minimal modifications of deep neural networks using verification," *23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, vol. 73, pp. 260–278, 2020.

[8] G. Dong, J. Sun, J. Wang, X. Wang, and T. Dai, "Towards Repairing Neural Networks Correctly," *IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, 2021.

[9] F. Fu and W. Li, "Sound and complete neural network repair with minimality and locality guarantees," *International Conference on Learning Representations (ICLR)*, 2022.

[10] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems," in *Computer Aided Verification*, Lecture Notes in Computer Science, pp. 3–17, Springer, 2020.

[11] J. Ferlez and Y. Shoukry, "Bounding the Complexity of Formally Verifying Neural Networks: A Geometric Approach," in *2021 60th IEEE Conference on Decision and Control*, pp. 5104–5109, 2021.