Safe Robot Learning in Assistive Devices through Neural Network Repair

Anonymous Author(s)
Affiliation
Address
email

Abstract: Assistive robotic devices are a particularly promising field of application for neural networks (NN) due to the need for personalization and hard-to-model human-machine interaction dynamics. However, NN based estimators and controllers may produce potentially unsafe outputs over previously unseen data points. In this paper, we introduce an algorithm for updating NN control policies to satisfy a given set of formal safety constraints, while also optimizing the original loss function. Given a set of mixed-integer linear constraints, we define the NN repair problem as a Mixed Integer Quadratic Program (MIQP). In extensive experiments, we demonstrate the efficacy of our repair method in generating safe policies for a lower-leg prosthesis.

Keywords:

1 Introduction

Robot learning has the potential to revolutionize the field of wearable robotic devices, such as prosthetics, orthoses and exoskeletons [1]. Often, such devices are designed in a "one size fits all" manner using models based on average population statistics. However, machine learning techniques can help adapt control parameters automatically to the wearer's individual characteristics, motion patterns or biomechanics. The result is a substantial improvement in ergonomic comfort and quality-of-life. Despite these benefits, the adoption of machine learning and, in particular, deep learning [2] in this field is still limited, largely due to safety concerns. In the case of a prosthesis, for example, it may be important to guarantee that the generated control values do not exceed a maximum threshold under a set of testing conditions. Other safety constraints include limits on velocities, joint angles and bounds on the change in control inputs.

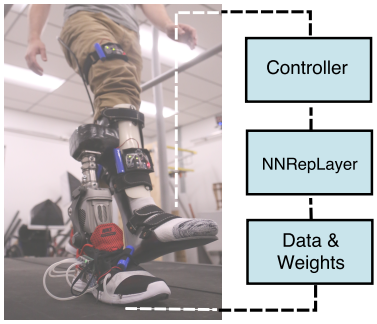

Figure 1: A lower-leg prosthesis running a neural network for control. A neural network repair process ensures that safety constraints are satisfied.

In this paper, we derive an algorithm for training neural networks controllers to satisfy given safety specifications, in addition to fitting the given training/test data. In particular, we use our approach to derive controllers for a robotic lower-leg prosthesis that satisfy basic safety conditions, (see Fig. 1). Our proposed approach inputs a trained network (e.g., a network obtained using backpropagation on the training data) along with a specification that places restrictions on the possible outputs for a given set of inputs. It then generates a modified set of weights that obeys the desired safety constraints on the output using deterministic global optimization. We provide theoretical guarantees of optimality for this technique (assuming no numerical errors). Furthermore, in real-world robot experiments we show that the introduced methodology produces safe neural policies for a lower-leg prosthesis satisfying a variety of constraints.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

Contributions: The proposed method can repair any layer in a network increasing the size of the solution space and the likelihood of a feasible successful repair. More importantly, it comes with theoretical guarantees on successfully repairing all discovered unsafe samples. When compared to retraining or fine-tuning methods, it also has two distinct benefits: (1) it does not require modification of the training data so as to satisfy the constraints, and (2) it does not utilize gradient optimization methods which do not guarantee constraint satisfaction even for the discovered unsafe data points. Finally, when compared to other property driven repair works, i.e., [3, 4, 5], it is applied for the first time to a real physical system, i.e., a powered prosthetic device, as opposed to a model.

Related Work Learning based control for prosthetics is motivated by the challenges in modeling human-prosthesis dynamics, which exhibits time varying behavior. Several different approaches that utilize learning based methods for controlling prosthetic devices have been proposed in the literature [6, 7]. Nevertheless, this short review focuses only on neural network (NN) based methods. Gao et al. [8] provide a prosthetic controller based on recurrent neural networks, and show that this type of controller effectively minimizes the difference between desired and actual trajectories on a powered prosthetic device. In another direction, Keleş and Yucesoy [9] and Vonsevych et al. [10] focus on utilizing electromyography (EMG) signals to predict control parameters for prosthetic ankles and hands. Our work follows a similar approach in predicting ankle control parameters with a neural network, which then drives the prosthesis through a PD controller. The verification of neural networks has been widely studied in order to check specifications for a given NN as a standalone component [11, 12, 13, 14] (for a survey see [15]). or as part of a closed loop system [16, 17, 18]. Testing techniques can produce counterexamples (adversarial samples) for a variety of NN-based applications, e.g., [19, 12, 20, 21, 22, 23]. One approach to ensuring that a NN satisfies given set of safety properties is to use retraining and fine-tuning based on counter-examples [24, 25]. However, this approach has a number of pitfalls. First, the labels for the counterexamples need to be available. This may involve further data-collection, which is often cumbersome. Also, gradient descent optimization approaches cannot provide any guarantees that the result satisfies the provided constraints. Our approach, in contrast, avoids generating adversarial examples or performing gradient descent. The problem of "repairing" a network, which involves modifying weights of the network in a minimal manner to satisfy some safety properties, is thus more involved than simply retraining a network on some additional data involving counterexamples. Three methods in the literature [3, 4, 5] make progress toward the goal of repairing NNs. Goldberger et al. [3] can only repair the output layer, which drastically reduces the space of possible successful repairs (and in many cases a repair is not even possible). The method of Fu and Li [4] produces patches for certain partitions of the input space wherein the corresponding outputs are modified. The direction of adding patches on the NN output is very promising; however, it is unclear how the approach will scale for high dimensional inputs, since it requires partitioning the input space into affine subregions. Finally, Dong et al. [26] primarily focus on counter-example generation, and the repairs are performed through retraining.

2 Problem Formulation

Without loss of generality, we motivate and discuss our approach using the task of learning safe robot controllers for a lower-leg prosthesis. The goal of this task is to learn a policy π_θ which generates control values for the ankle angle of the powered prosthesis given a set of sensor values. Most critically, however, policy π_θ is required to satisfy a set of safety constraints Ψ. Fig. 2 provides an overview of our methodology in addressing this challenge. For now, we assume that an unsafe prior policy network may exist, see Fig. 2 (left). The network parameters θ may be learned with through imitation learning [8], reinforcement learning [27] or any of the common machine learning paradigms. The policy may be optimized for task efficiency, e.g., stable and low-effort walking gaits, but may not yet satisfy any safety constraints. Our goal is to find an adjusted set of network parameters that satisfy any such constraint. We may now choose, for example, to restrict the control rate to specific bounds. Running the network on the input values of a validation data set reveals that violations occur at a number of time steps, as seen in Fig. 2. Our approach, called **NNRepLayer**

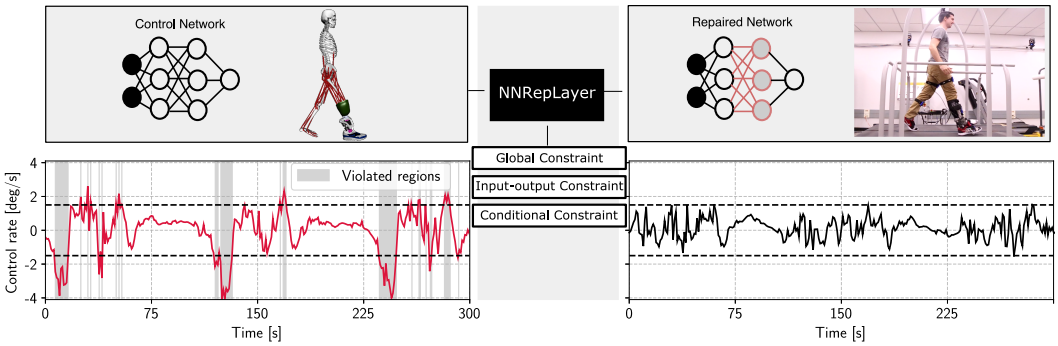

Figure 2: Overview of our approach. Left: a given (unsafe) neural network is trained to control a prosthesis. However, its outputs violate the formal safety constraints. Right: Using our NN Repair strategy, we identify an adjusted set of neural network weights that removes all violations while still maintaining the underlying behavior of the controller.

(Layer-wise Neural Network Repair), takes the original network parameters θ and the predicates Ψ and yields the updated parameters that generate no violations.

Notation. We denote the set of variables $\{a_1, a_2, \cdots, a_N\}$ with $\{a_n\}_{n=1}^N$. Let π_θ be a network policy with L hidden layers. The nodes at each layer $l \in \{l\}_{l=0}^L$ are represented by x^l, where $|x^l|$ denotes the dimension of layer l (x^0 represents the input of network). The network's output is also denoted by y or $\pi_\theta(x^0)$. We consider fully connected policy networks with weight and bias terms $\{(\theta_w^l, \theta_b^l)\}_{l=1}^{L+1}$. The training data set of N inputs x_n^0 and target outputs t_n is denoted by $\{(x_n^0, t_n)\}_{n=1}^N$ sampled from the input-output space $\mathcal{X} \times \mathcal{T} \subseteq \mathbb{R}^{|x^0|} \times \mathbb{R}^{|t|}$. We use x_n^l to denote the vector of nodes at layer l for sample n. In this work, we focus on the policy networks with the Rectified Linear Unit (ReLU) activation function $R(z) = \max\{0, z\}$. Thus, given the nth sample, in the lth hidden layer, we have $x^l = R\left(\theta_w^l x^{l-1} + \theta_b^l\right)$. An activation function is not applied to the last layer, i.e. $y = \theta_w^{L+1} x^L + \theta_b^{L+1}$.

Problem Statement (Repair Problem). Let π_θ be a trained policy network over the training input-output space $\mathcal{X} \times \mathcal{T} \subseteq \mathbb{R}^{|^0|} \times \mathbb{R}^{|t|}$ and $\Psi(y, x^0)$ be a predicate on the output y of network for a set of inputs of interest $x^0 \in \mathcal{X}_r \subseteq \mathcal{X}$. The Repair Problem is to modify the weight θ_w and bias terms θ_b of π_θ such that the repaired policy π_{θ_r} satisfies $\Psi(y, x^0)$.

The repair of the policy network should not only satisfy the predicate $\Psi(y, x^0)$ but should also maintain the performance of original policy. To satisfy the latter, the method proposed in [3] ensures the satisfaction of predicate only with a minimal deviation of weights in the last layer. However, as we show latter in the experimental results, the repair of last layer is not necessarily feasible or sufficient to satisfy the predicates with a minimal deviation from the original parameters. Moreover, the minimal deviation from the original weights is not a sufficient guarantee to maintain the original performance of network. It is well-known that subtle changes in the weights may cause the network to significantly deviate from its original performance [28]. Therefore, it is important for the repaired policy π_{θ_r} to also minimize the loss w.r.t. its original training data. We propose NNRepLayer (Layer-wise Neural Network Repair) that satisfies a predicate $\Psi(y, x^0)$ by repairing a specific layer of the policy network while minimizing the training loss.

3 NNRepLayer

We can formulate our framework as the minimization problem of the loss function $E(\theta_w, \theta_b)$ subject to $(x^0, t) \in \mathcal{X} \times \mathcal{T}$ and $\Psi(y, x^0)$ for $x^0 \in \mathcal{X}_r$. However, the resulting optimization problem is non-convex and difficult to solve due to the nonlinear ReLU activation function and high-order nonlinear constraints resulted from the multiplication of terms involving the weight/bias variables.

In our approach, we obtain a sub-optimal solution by just focusing on repairing a single layer. We therefore modify the weight and bias terms of a single layer to adjust the predictions so as to minimize $E(\theta_w, \theta_b)$ and to satisfy $\Psi(y, x^0)$. Thus, we solve the following problem

Problem 1. *Let π_θ denote a trained policy network with L hidden layers over the training input-output space $\mathcal{X} \times \mathcal{T} \subseteq \mathbb{R}^{|x^0|} \times \mathbb{R}^{|t|}$ and $\Psi(y, x^0)$ denote a predicate representing constraints on the output y of π_θ for the set of inputs of interest $x^0 \in \mathcal{X}_r \subseteq \mathcal{X}$. NNRepLayer modifies the weights of a layer $l \in \{1, \cdots, L+1\}$ in π_θ such that the new network π_{θ_r} satisfies $\Psi(y, x^0)$ while minimizing the loss of network $E(\theta_w^l, \theta_b^l)$ with respect to its original training set.*

Since $\mathcal{X}_r$ and $\mathcal{X}$ are not necessarily convex, we formulate NNRepLayer over a data set $\{(x_n^0, t_n)\}_{n=1}^N \sim \mathcal{X} \times \mathcal{T} \cup \mathcal{X}_r \times \tilde{\mathcal{T}}$, where $\tilde{\mathcal{T}}$ is the set of original target values of inputs in $\mathcal{X}_r$. The predicate $\Psi(x^0, y)$ defined over $x^0 \in \mathcal{X}_r$ is not necessarily compatible with the target values in $\tilde{\mathcal{T}}$. It means that the predicate may bound the NN output for $\mathcal{X}_r$ input space such that not allowing an input $x^0 \in \mathcal{X}_r$ to reach its target value in $\tilde{\mathcal{T}}$. It is a natural constraint in many applications. For instance, due to the safety constraints, we may not allow a NN controller to follow its original control reference for a given unsafe set of input states. For a given layer l, we also define $E(\theta_w^l, \theta_b^l)$ in the form of sum of square loss $E(\theta_w^l, \theta_b^l) = \sum_{n=1}^N \|y_n(x_n^0, \theta_w^l, \theta_b^l) - t_n\|_2^2$, where $\|\cdot\|_2$ denotes the Euclidean norm. Here, since we only repair the weight and bias terms of target layer l, the loss term E is a function of θ_w^l and θ_b^l, respectively. Hence, the weight and bias terms of all layers except the target layer l are fixed. We define our optimization formulation as follows.

NNRepLayer Optimization Formulation. Let π_θ be a neural network with L hidden layers, $\Psi(y, x^0)$ be a predicate, and $\{(x_n^0, t_n)\}_{n=1}^N$ be an input-output data set sampled from $(\mathcal{X} \times \mathcal{T}) \cup (\mathcal{X}_r \times \tilde{\mathcal{T}})$ over the sets $\mathcal{X}$, $\mathcal{X}_r$, $\mathcal{T}$, and $\tilde{\mathcal{T}}$ all as defined in Problem 1. NNRepLayer minimizes the loss (1) by modifying θ_w^l and θ_b^l subject to the constraints (2)-(5).

$$
\begin{array}{lll}
(1) & \min\limits_{\substack{\theta_w^l, \theta_b^l, y_n, \delta, \\ \{x_n^i\}_{i=l}^L, \{s^i\}_{i=l}^L, \{\phi_n^i\}_{i=l}^L}} & E(\theta_w^l, \theta_b^l) + \delta, \\
 & \text{s.t.} & \\
(2) & y_n = \theta_w^{L+1} x_n^L + \theta_b^{L+1}, & \\
(3) & x_n^i = R(\theta_w^i x_n^{i-1} + \theta_b^i), & \text{for } \{i\}_{i=l}^L \\
(4) & \Psi(y_n, x_n^0), & \text{for } x_n^0 \in \mathcal{X}_r \\
(5) & \delta \geq \|\theta_w^l - \theta_w^{l,init}\|_\infty, \ \|\theta_b^l - \theta_b^{l,init}\|_\infty \geq 0. &
\end{array}
$$

Here, constraint (2) represents the linear forward pass of network's last layer. Constraint (3) represents the forward pass of hidden layers starting from the layer l. Except the weight and bias terms θ_w and θ_b in the lth layer, the weight and bias terms of the subsequent layers $\{(\theta_w^l, \theta_b^l)\}_{l=1}^{L+1}$ are fixed. The sample values of x_n^{l-1} can be obtained by the weighted sum of the nodes in its previous layers starting from x_n^0 for all N samples $\{n\}_{n=1}^N$. The ReLU activation functions $R(z) = \max\{0, z\}$ are modeled with a disjunctive inequality constraint $[\phi = 0, x \leq 0] \vee [\phi = 1, s \leq 0]$. Assume ϕ is a Boolean variable and $z = x - s$, where $x, s \geq 0$. We have $x = z$ when $\phi = 1$ ($z \geq 0$) and $x = 0$ when $\phi = 0$ ($z = -s < 0$). Here, x determines the output of ReLU activation function $R(z)$, the disjunctive inequalities are then relaxed as mixed integer algebraic equations by the Big-M method [29].

Constraint (4) is a given predicate on y defined over $x^0 \in \mathcal{X}_r$. NNReplayer addresses the predicates of the form $\bigvee_{c=1}^C \psi_c(x^0, y)$ where C represents the number of disjunctive propositions and ψ_i is an affine function of x^0 and y. Finally, constraint (5) bounds the entry-wise max-norm error between the weight and bias terms θ_w^l and θ_b^l, and the original $\theta_w^{l,init}$ and $\theta_b^{l,init}$ by δ.

Considering the quadratic loss function $E(\theta_w^l, \theta_b^l)$ and the affine disjunctive forms of $\Psi(y_n, x_n^0)$ and $R(\theta_w^i x_n^{i-1} + \theta_b^i)$, we solve NNRepLayer as a Mixed Integer Quadratic Program (MIQP).

Theorem 1. *Given the predicate $\Psi(y, x^0)$, and the input-output data set $\{(x_n^0, t_n)\}_{n=1}^N$ sampled from $(\mathcal{X} \times \mathcal{T}) \cup (\mathcal{X}_r \times \tilde{\mathcal{T}})$ over the sets $\mathcal{X}$, $\mathcal{X}_r$, $\mathcal{T}$, and $\tilde{\mathcal{T}}$ as defined in Problem 1, assume that θ_w^l and θ_b^l are feasible solutions to NNRepLayer. Then, for all input samples x_n^0 from $\{(x_n^0, t_n)\}_{n=1}^N$, $\Psi(\pi_{\theta_r}(x_n^0), x_n^0)$ is satisfied.*

Proof. Since the feasible solutions θ_w^l and θ_b^l satisfy the hard constraint (4) for the repair data set $\{(x_n^0, t_n)\}_{n=1}^N$, $\Psi(\pi_{\theta_r}(x_n^0), x_n^0)$ is satisfied. □

Given Thm. 1, the following Corollary is straightforward.

Corollary 1. *Given the predicate $\Psi(y, x^0)$, and the input-output data set $\{(x_n^0, t_n)\}_{n=1}^N$ sampled from $(\mathcal{X} \times \mathcal{T}) \cup (\mathcal{X}_r \times \tilde{\mathcal{T}})$ over the sets $\mathcal{X}$, $\mathcal{X}_r$, $\mathcal{T}$, and $\tilde{\mathcal{T}}$ as defined in Problem 1, assume that $\theta_w^{l^*}$ and $\theta_b^{l^*}$ are the optimal solutions to the NNRepLayer. Then, for all input samples x_n^0 from $\{(x_n^0, t_n)\}_{n=1}^N$, $\Psi(\pi_{\theta_r}(x_n^0), x_n^0)$ is satisfied.*

4 Evaluation

We explore the applicability of the framework in satisfying the following three types of constraints. **Global constraints** that encode global bounds on the network's output, i.e., $y \in [y_{min}, y_{max}]$. **Input-output constraints** that ensure the network's output y to stay within a certain bound with respect to the network's input x^0, i.e., $\{\psi_c(x^0, y) \leq 0\}_{c=0}^C$, where C is the number of constraints and ψ_c is an affine function of x^0 and y. Finally, **conditional constraints** that encode if-then-else constraints described as $\{\psi_c(x^0, y) \leq 0, \text{if } x^0, y \in S_c\}_{c=0}^C$, where C specifies the number of conditions, ψ_c is an affine function of x^0 and y, and $S_c \subseteq \mathcal{X} \times \mathcal{T}$. We designed a number of experiments to validate that our repair framework can successfully apply these constraints to the policy network. Following our motivation, all experiments were performed on the prosthetic walking gait generation task introduced in Fig. 2. Through these experiments we aim to answer the following questions: (1) Does our method enable the Prosthetic device to address all the three types of aforementioned constraints? (2) Can the repaired controller be employed in a real walking scenario successfully? (3) How robust is the policy repaired through our technique against the unseen constraint-violating samples?

Experimental Setup. In our experiment, we train a policy network π_θ for controlling a prosthesis, which then undergoes the repair process to ensure compliance with the safety constraints. To this end, we first train the model using an imitation learning [30] strategy. For data collection, we conducted a study approved by the Institutional Review Board (IRB), in which we recorded the walking gait of a healthy subject without any prosthesis. Walking data included three inertial measurement units (IMUs) mounted via straps to the upper leg (Femur), lower leg (Shin), and foot. The IMUs acquired both the angle and angular velocity of each limb portion in the world coordinate frame at 100Hz. Ankle angle α_a was calculated as a post process from the foot and lower limb IMUs. We then trained the NN to generate the ankle angle from upper and lower limb IMU sensor values.

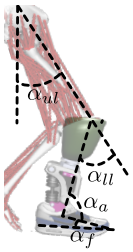

Figure 3: Prosthetic device model.

More specifically, the NN model receives the angle and velocity from the upper and lower limb sensors, $\alpha_{ul}, \dot{\alpha}_{ul}, \alpha_{ll}, \dot{\alpha}_{ll}$, respectively, to predict the ankle angle α_a which is, later, used as the control parameter for a PD controller on the prosthetic. See Fig. 3 for a visualization of the individual sensor readings. We used a sliding window of input variables, denoted as dt ($dt = 10$ in all our experiments), to account for the temporal influence on the control parameter and to accommodate for noise in the sensor readings. Therefore, the input to the network is $dt \times |x^0|$, or more specifically the current and previous dt sensor readings. In all experiments, we trained a three-hidden-layer deep policy network with 32 ReLU nodes at each hidden layer. After the networks were fully trained we assessed the policy for constraint violations and collected samples for NNRepLayer. We tested NNRepLayer on the last and the second to the last layer of network policy to satisfy the constraints with a subset of the original training data including both adversarial and non-adversarial samples. In all experiments, we used 150 samples in NNRepLayer and a held out set of size 2000 for the testing. Finally, the repaired policies to satisfy global and input-output constraints are tested on a prosthetic device for 10 minutes of walking, see Fig. 5. More specifically, the same healthy subject was fitted with an ankle bypass; a carbon fiber structure molded to the lower limb and constructed such that

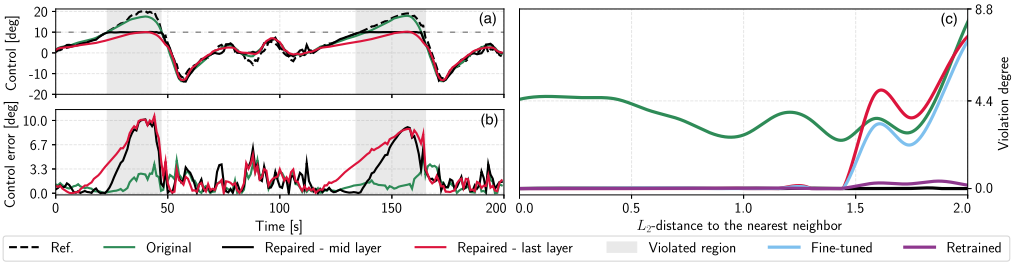

Figure 4: Global constraint: (a) ankle angle, α_a, (b) the error between the predicted and the reference controls, (c) the violation degree vs. L_2-distance between the test and repair sample inputs.

a prosthetic ankle can be attached to allow the able-bodied subject to walk on the prosthesis. The participant is then asked to walk again for 10 minutes to assess whether constraints are satisfied.

To evaluate if the repair can be generalized to the unseen adversarial samples, we analyze the *violation degree*. The violation degree is measured as the distance of the network output with respect to the constraint set. For each experiment, we explain how this distance between the output and the constraint set is calculated. We compared our framework with fine-tuning [24] and retraining. Adversarial samples in the repair data set are hand-labeled for fine-tuning and retraining so that the target outputs satisfy the given predicates. In fine-tuning, as proposed in [24, 31], we used the collected adversarial data set to train all the parameters of the original policy by gradient descent using a small learning rate (10^{-4}). To avoid over-fitting to the adversarial data set, we trained the weights of the top layer first, and thereafter fine-tuned the remaining layers for a few epochs. The same hand-labeling strategy is applied in retraining, except that a new policy is trained from scratch for all original training samples. In both methods, we trained the policy until all the adversarial samples in the repair data set satisfy the given predicates on the network's output.

4.1 Experiments and Results

Global Constraint. The global constraint ensures that the prosthesis control, i.e., α_a, stays within a certain range and never outputs an unexpected large value that disturbs the user's walking balance. Additionally, the prosthetic device we utilize in these scenarios contains a parallel compliant mechanism. As such, either the human subject or the robotic controller could potentially drive the mechanism into the hard limits, potentially damaging the device. In our walking tests, we therefore specified global constraints such that the ankle angle stays within the bounds of $[-14, 24]$ [deg] regardless of whether it is driven by the human or the robot. In simulation experiments, we enforced artificially strict bounds on the ankle angle α_a to never exceed $\alpha_a = 10$ [deg] which is a harder bound to satisfy. We defined the degree of violation as 0 if $\alpha_a \in [\alpha_a^{min}, \alpha_a^{max}]$, and $min\{|\alpha_a - \alpha_a^{max}|, |\alpha_a - \alpha_a^{min}|\}$, otherwise. As shown in Fig. 4 (a), the repaired network can successfully satisfy the constraints in the previously violating regions while maintaining the tracking performance of the controller in the unconstrained regions. Figure 4 (c) demonstrates that the violation degree after repairing the mid layer stays almost zero for even distant originally violating data points. The red control signal in Fig. 5 also shows that our method successfully imposes the control bounds $[-14, 24]$ to the prosthesis walking.

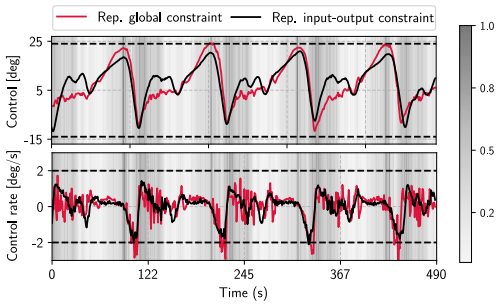

Figure 5: Real prosthesis walking test results for imposing the global constraint of $[-14, 24]$ to the control (shown in red) and bounding the control rate by 2 [deg/s] (shown in black). The color bar represents the normalized L_2-distance of each test input to its nearest neighbor in the repair set.

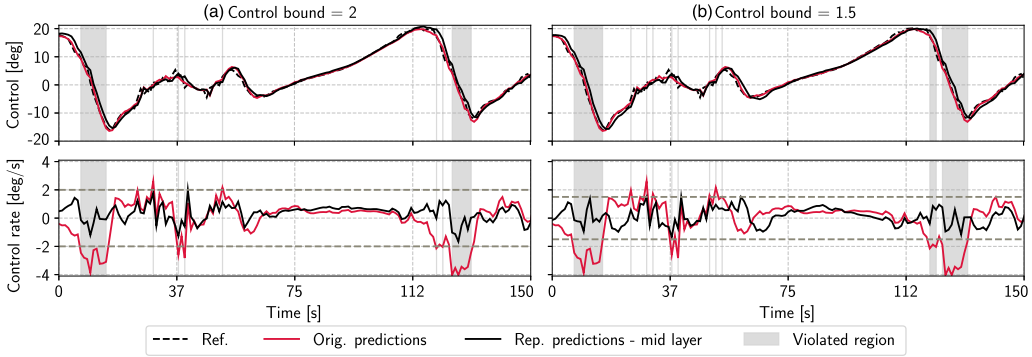

Figure 7: Input-output constraint: Ankle angles and Ankle angle rates for bounds (a) $\Delta\alpha_a = 2$ and (b) $\Delta\alpha_a = 1.5$

Input-output Constraint. Deep neural networks as highly non-linear function approximators have the ability to change the outputs more rapidly than what is feasible for the robotic prosthesis or for the human subject to accommodate. Therefore, we propose an additional constraint over the possible change of control actions from one time-step to the next. This constraint should act to both smooth the control action in the presence of sensor noise, as well as to reduce hard peaks and oscillations in the control action. To capture this constraint as an input-output relationship, we trained the policy network by adding the previous dt control actions $\{\alpha_a(i)\}_{i=t-dt}^{t-1}$ as inputs to the policy network along the values of upper and lower limb sensors. Imposed constraints in this example follow the form $|\alpha_a(t) - \alpha_a(t-1)| \leq \Delta\alpha_a^{max}$. In prosthetic walking tests, we bounded the control rate by $\Delta\alpha_a^{max} = 2$ [deg/s], and in our simulations we tested $\Delta\alpha_a^{max} = 1.5$ and $\Delta\alpha_a^{max} = 2$ [deg/s]. Our simulation results in Fig. 7 demonstrate that NNRepLayer satisfies both bounds on the control rate which subsequently results in a smoother control output. It can also be observed that NNRepLayer also successfully preserves the tracking performance of controller. In this experiment, we defined the violation degree as 0 if the $|\Delta\alpha_a(t)| \leq \Delta\alpha_a^{max}$ and $|\Delta\alpha_a(t) - \Delta\alpha_a^{max}|$, otherwise. Figure 6 (a) demonstrates that the violation degree of NNRepLayer is almost zero by increasing the distance of the test samples from the repair set. Same results are obtained in the prosthetic walking tests as shown in Fig. 5. The satisfaction of input-output constraints (black signal) is achieved even for the samples that are distant from the repair set. Finally, in this experiment, we observed that applying NNRepLayer to the last layer does not obtain even a feasible solution to the optimization problem (1).

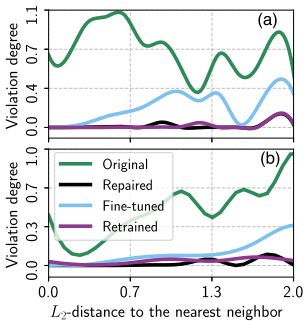

Figure 6: The violation degree vs. L_2-distance between the test and repair sample inputs for (a) input-output constraint, and (b) conditional constraint cases.

Conditional Constraint. Depending on the ergonomic needs and medical history of a patient, the attending orthopedic doctor or prosthetist may identify certain body configurations that are harmful, e.g., they may increase the risk of osteoarthritis or musculoskeletal conditions [32, 33]. Following this rationale, we define a region $\mathcal{S}$ of joint angles space that should be avoided. An example of such a region is demonstrated in Fig. 8 as a grey box $\mathcal{S} = \{(\alpha_{ul}, \alpha_a) \mid \alpha_{ul} \in [-2, -0.5], \alpha_a \in [1, 3]\}$ in the joint space of ankle and femur angles. To satisfy this constraint the control rate should be tuned such that the joint ankle and femur angles stay out of set $\mathcal{S}$. This constraint can be defined as an if-then-else proposition $\alpha_{ul} \in [-2, -0.5] \implies \big(\alpha_a \in (-\infty, 1]\big) \vee \big(\alpha_a \in [3, \infty)\big)$ which can be formulated as the disjunction of linear inequalities on the network's output. For each given test input and its corresponding output α_a, the degree of violation is defined as 0 if the corresponding output is inside the box and its distance to the box otherwise.

Table 1: The table reports the Mean Absolute Error (MAE) between the repaired and the original outputs, the percentage of adversarial samples that are repaired, also called the Repair Efficacy, and the runtime of NNRepLayer for each constraint case. The metrics for fine-tuning and retraining are the average of 50 runs. For fixed repair samples, NNRepLayer returns the same outcome each run.

	MAE			Repair Efficacy [%]			Runtime [s]		
	Fine-tune	Retrain	NNRepLayer	Fine-tune	Retrain	NNRepLayer	Fine-tune	Retrain	NNRepLayer
Global	1.2 ± 0.03	1.4 ± 0.08	1.3	97 ± 4	98 ± 3	**99**	25 ± 13	127 ± 30	414
Input-output	0.6 ± 0.03	0.5 ± 0.04	0.5	88 ± 2	98 ± 1	**99**	8 ± 2	101 ± 1	90
Conditional	0.7 ± 0.10	0.31 ± 0.03	0.38	72 ± 5	76 ± 2	**91**	18 ± 3	180 ± 2	555

Figure 8 demonstrates the output of new policy after repairing with NNRepLayer. As it is shown, our method avoids the joint ankle and femur angles to enter the unsafe region S. Figure 6 (b) also illustrates low output violation degree for the distant test input samples from the repair input set. Finally, we also observed that repairing the last layer does not result in a feasible solution.

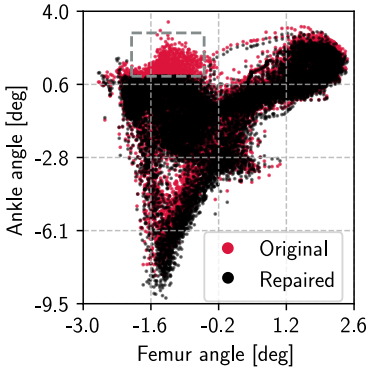

Figure 8: NNRepLayer enforcing the conditional constraints to keep the joint femur-ankle angles out of the grey box.

Comparison w\Fine-tuning and Retraining. In each experiment, we demonstrated the violation degree of our method compared with fine-tuning and retraining. Table 1 better illustrates the success of our method in satisfying the constraints while maintaining the control performance. As shown in Table 1, retraining and NNRepLayer both perform well in maintaining the minimum absolute error and the generalization of constraint satisfaction to the unseen testing samples for global and input-output constraints. However, according the Table 1 the satisfaction of if-then-else constraints is challenging for retraining and fine-tuning as the repair efficacy is dropped by almost 30% using these techniques. It also shows the power of our technique in generalizing the satisfaction of conditional constraints to the unseen cases.

5 Limitations & Conclusions

In this paper, we introduced an algorithm for training safe neural network controllers that satisfy a formal set of safety constraints. Our approach, NNRepLayer, performs a global optimization step in order to perform layer-wise repair of neural network weights. In real-robot experiments, we have shown that the introduced methodology produces safe neural policies for a lower-leg prosthesis satisfying a variety of constraints. We argue that this type of approach is critical for human-centric and safety-critical applications of robot learning, e.g., the next-generation of assistive robotics.

Limitations: The introduced approach does not generally ensure that for *any* input the constraints will be satisfied. Instead it guarantees this property for all data points provided at time of repair. Hence, proper care has to be taken to ensure that the repair process involves representative samples of the variety of inputs seen in the application domain. From a computational vantage point, solving the MIQP underlying NNRepLayer is a demanding process which scales with the size of the network. In our experiments, we successfully repaired NN layers with up to 64 neurons, with global optimization taking between multiple minutes and up to 6 hours. Similarly, our approach is limited to repairing individual layers in a network. Early results on iteratively repairing multiple layers are promising and will be reported in the future. However, our approach cannot simultaneously repair multiple layers or the entire network. This restricts its applicability, at the moment, to control tasks with low- to medium-sized dimensionality, i.e., repair cannot be performed for layer with thousands of neurons.

References

[1] M. Kyrarini, F. Lygerakis, A. Rajavenkatanarayanan, C. Sevastopoulos, H. R. Nambiappan, K. K. Chaitanya, A. R. Babu, J. Mathew, and F. Makedon. A survey of robots in healthcare. *Technologies*, 9(1):8, 2021.

[2] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

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

[4] F. Fu and W. Li. Sound and complete neural network repair with minimality and locality guarantees. In *10th International Conference on Learning Representations (ICLR)*, 2021.

[5] X. Yang, T. Yamaguchi, H.-D. Tran, B. Hoxha, T. T. Johnson, and D. Prokhorov. Neural network repair with reachability analysis. *arXiv preprint arXiv:2108.04214*, 2021.

[6] N. Thatte, N. Srinivasan, and H. Geyer. Real-time reactive trip avoidance for powered transfemoral prostheses. In *Robotics Systems and Sciences (RSS)*, 2019.

[7] G. Clark, J. Campbell, and H. B. Amor. Learning predictive models for ergonomic control of prosthetic devices. In J. Kober, F. Ramos, and C. Tomlin, editors, *Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 2268–2278. PMLR.

[8] C. Gao, R. Gehlhar, A. D. Ames, S.-C. Liu, and T. Delbruck. Recurrent neural network control of a hybrid dynamical transfemoral prosthesis with edgedrnn accelerator. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5460–5466. IEEE, 2020.

[9] A. D. Keleş and C. A. Yucesoy. Development of a neural network based control algorithm for powered ankle prosthesis. *Journal of Biomechanics*, 113:110087, 2020.

[10] K. Vonsevych, M. F. Goethel, J. Mrozowski, J. Awrejcewicz, and M. Bezuglyi. Fingers movements control system based on artificial neural network model. *Radioelectronics and Communications Systems*, 62(1):23–33, 2019.

[11] R. Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *15th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pages 269–286. Springer, 2017.

[12] V. Tjeng, K. Y. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations (ICLR)*, 2019.

[13] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Output range analysis for deep feedforward neural networks. In *10th International Symposium NASA Formal Methods (NFM)*, volume 10811 of *LNCS*, pages 121–138. Springer.

[14] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pages 443–452. Springer, 2019.

[15] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer. Algorithms for verifying deep neural networks. 4(3-4):244–404. ISSN 2167-3888.

[16] X. Sun, H. Khedr, and Y. Shoukry. Formal verification of neural network controlled autonomous systems. In *22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019.

[17] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verifying the safety of autonomous systems with neural network controllers. 20(1). ISSN 1539-9087.

[18] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Learning and verification of feedback control systems using feedforward neural networks. In *Analysis and Design of Hybrid Systems*, 2018.

[19] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In R. Majumdar and V. Kunčak, editors, *29th International Conference on Computer Aided Verification (CAV)*, volume 10426 of *LNCS*, pages 3–29. Springer, 2017.

[20] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *40th International Conference on Software Engineering (ICSE)*, 2018.

[21] T. Dreossi, A. Donze, and S. A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. 63:1031–1053, 2019.

[22] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski. Requirements-driven test generation for autonomous vehicles with machine learning components. *IEEE Transactions on Intelligent Vehicles*, 5:265–280, 2020. doi:10.1109/TIV.2019.2955903.

[23] T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Counterexample-guided data augmentation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*, pages 2071–2078, 2018.

[24] A. Sinitsin, V. Plokhotnyuk, D. Pyrkin, S. Popov, and A. Babenko. Editable neural networks. *arXiv preprint arXiv:2004.00345*, 2020.

[25] X. Ren, B. Yu, H. Qi, F. Juefei-Xu, Z. Li, W. Xue, L. Ma, and J. Zhao. Few-shot guided mix for dnn repairing. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 717–721. IEEE, 2020.

[26] G. Dong, J. Sun, X. Wang, X. Wang, and T. Dai. Towards repairing neural networks correctly. In *IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, pages 714–725, 2021.

[27] X. Gao, J. Si, Y. Wen, M. Li, and H. H. Huang. Knowledge-guided reinforcement learning control for robotic lower limb prosthesis. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 754–760, 2020. doi:10.1109/ICRA40945.2020.9196749.

[28] X. Zeng and D. Yeung. Sensitivity analysis of multilayer perceptron to input and weight perturbations. *IEEE Transactions on Neural Networks*, 12(6):1358–1366, 2001. doi:10.1109/72.963772.

[29] P. Belotti, L. Liberti, A. Lodi, G. Nannicini, A. Tramontani, et al. Disjunctive inequalities: applications and extensions. *Wiley Encyclopedia of Operations Research and Management Science*, 2:1441–1450, 2011.

[30] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3 (6):233–242, 1999.

[31] V. Taormina, D. Cascio, L. Abbene, and G. Raso. Performance of fine-tuning convolutional neural networks for hep-2 image classification. *Applied Sciences*, 10(19):6940, 2020.

[32] D. C. Morgenroth, A. C. Gellhorn, and P. Suri. Osteoarthritis in the disabled population: a mechanical perspective. *PM&R*, 4(5):S20–S27, 2012.

[33] A. Ekizos, A. Santuz, A. Schroll, and A. Arampatzis. The maximum lyapunov exponent during walking and running: Reliability assessment of different marker-sets. *Frontiers in Physiology*, 9:1101, 2018. ISSN 1664-042X.