Relational DNN Verification With Cross Executional Bound Refinement

Debangshu Banerjee [1] **Gagandeep Singh** [1 2]

Abstract

We focus on verifying relational properties defined over deep neural networks (DNNs) such as robustness against universal adversarial perturbations (UAP), certified worst-case hamming distance for binary string classifications, etc. Precise verification of these properties requires reasoning about multiple executions of the same DNN. However, most of the existing works in DNN verification only handle properties defined over single executions and as a result, are imprecise for relational properties. Though few recent works for relational DNN verification, capture linear dependencies between the inputs of multiple executions, they do not leverage dependencies between the outputs of hidden layers producing imprecise results. We develop a scalable relational verifier RACoon that utilizes cross-execution dependencies at all layers of the DNN gaining substantial precision over SOTA baselines on a wide range of datasets, networks, and relational properties.

1. Introduction

Deep neural networks (DNNs) have gained widespread prominence across various domains, including safety-critical areas like autonomous driving (Bojarski et al., 2016) or medical diagnosis (Amato et al., 2013), etc. Especially in these domains, the decisions made by these DNNs hold significant importance, where errors can lead to severe consequences. However, due to the black-box nature and highly nonlinear behavior of DNNs, reasoning about them is challenging. Despite notable efforts in identifying and mitigating DNN vulnerabilities (Goodfellow et al., 2014; Madry et al., 2018; Moosavi-Dezfooli et al., 2017; Potdevin et al., 2019; Wu et al., 2023; Sotoudeh & Thakur, 2020), these methods cannot guarantee safety. Consequently, significant research has been dedicated to formally verifying the safety properties of DNNs. Despite advancements, current DNN verification techniques can not handle relational properties prevalent in practical scenarios. Most of the existing efforts focus on verifying the absence of input-specific adversarial examples within the local neighborhood of test inputs. However, recent studies (Li et al., 2019a) highlight the impracticality of attacks targeting individual inputs. In practical attack scenarios (Liu et al., 2023; Li et al., 2019b;a), there is a trend towards developing universal adversarial perturbations (UAPs) (Moosavi-Dezfooli et al., 2017) designed to affect a significant portion of inputs from the training distribution. Since the same adversarial perturbation is applied to multiple inputs, the executions on different perturbed inputs are related, and exploiting the relationship between different executions is important for designing precise relational verifiers. Existing DNN verifiers working on individual executions lack these capabilities and as a result, lose precision. Beyond UAP verification, other relevant relational properties include measuring the worst-case hamming distance for binary string classification and bounding the worst-case absolute difference between the original number and the number classified using a digit classifier where inputs perturbed with common perturbation.

Key challenges: For precise relational verification, we need scalable algorithms to track the relationship between DNN's outputs across multiple executions. Although it is possible to exactly encode DNN executions with piecewise linear activation functions (e.g. ReLU) over input regions specified by linear inequalities as MILP (Mixed Integer Linear Program), the corresponding MILP optimization problem is computationally expensive. For example, MILP encoding of k executions of a DNN with n_r ReLU activations in the worst case introduces $O(n_r \times k)$ integer variables. Considering the cost of MILP optimization grows exponentially with the number of integer variables, even verifying small DNNs w.r.t a relational property defined over k execution with MILP is practically infeasible. For scalability, (Khedr & Shoukry, 2023) completely ignores the dependencies across executions and reduces relational verification over k executions into k individual verification problems solving them independently. SOTA relational verifier (Zeng et al., 2023) first obtains provably correct linear approximations of the DNN with existing non-relational verifier (Xu et al., 2020) without tracking any cross-execution dependencies then adds linear constraints at the input layer capturing

[*]Equal contribution [1]Department of Computer Science, University of Illinois Urbana-Champaign, USA [2]VMware Research, USA. Correspondence to: Debangshu Banerjee <db21@illinois.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

linear dependencies between inputs used in different executions. In this case, ignoring cross-execution dependencies while computing provably correct linear approximations of the DNN for each execution leads to the loss of precision (as confirmed by our experiments in Section 6). This necessitates developing scalable algorithms for obtaining precise approximations of DNN outputs over multiple executions that benefit from cross-execution dependencies.

Our contributions: We make the following contributions to improve the precision of relational DNN verification:

- In contrast to the SOTA baselines, we compute a provably correct parametric linear approximation of the DNN for each execution using parametric bounds of activation functions (e.g. ReLU) as done in existing works (Xu et al., 2021; Salman et al., 2019). Instead of learning the parameters for each execution independently as done in (Xu et al., 2021), we refine the parametric bounds corresponding to multiple executions together. In this case, the bound refinement at the hidden layer takes into account the cross-execution dependencies so that the learned bounds are tailored for verifying the specific *relational* property.
- For scalable cross-executional bound refinement, we (a) formulate a linear programming-based relaxation of the relational property, (b) find a provably correct differentiable closed form of the corresponding Dual function that preserves dependencies between parameters from different executions while being suitable for scalable differentiable optimization techniques, (c) using the differentiable closed form refine the parametric bound with scalable differential optimization methods (e.g. gradient descent).
- We develop RACoon (**R**elational DNN **A**nalyzer with **C**ross-Excuti**o**nal B**o**und Refi**n**ement) that formulates efficiently optimizable MILP instance with cross-executional bound refinement for precise relational verification.
- We perform extensive experiments on popular datasets, multiple DNNs (standard and robustly trained), and multiple relational properties showcasing that RACoon significantly outperforms the current SOTA baseline.[1]

2. Related Works

Non-relational DNN verifiers: DNN verifiers are broadly categorized into three main categories - (i) sound but incomplete verifiers which may not always prove property even if it holds (Gehr et al., 2018; Singh et al., 2018; 2019b;a; Zhang et al., 2018; Xu et al., 2020; 2021), (ii) complete verifiers that can always prove the property if it holds (Wang et al., 2018; Gehr et al., 2018; Bunel et al., 2020a;b; Bak et al., 2020; Ehlers, 2017; Ferrari et al., 2022; Fromherz et al., 2021; Wang et al., 2021; Palma et al., 2021; Anderson et al., 2020; Zhang et al., 2022a) and (iii) verifiers with probabilistic guarantees (Cohen et al., 2019; Li et al., 2022).

[1]Code at https://github.com/uiuc-focal-lab/RACoon

Relational DNN verifier: Existing DNN relational verifiers can be grouped into two main categories - (i) verifiers for properties (UAP, fairness, etc.) defined over multiple executions of the same DNN, (Zeng et al., 2023; Khedr & Shoukry, 2023; BANERJEE et al.), (ii) verifiers for properties (local DNN equivalence (Paulsen et al., 2020)) defined over multiple executions of different DNNs on the same input (Paulsen et al., 2020; 2021). For relational properties defined over multiple executions of the same DNN the existing verifiers (Khedr & Shoukry, 2023) reduce the verification problem into L_∞ robustness problem by constructing product DNN with multiple copies of the same DNN. However, the relational verifier in (Khedr & Shoukry, 2023) treats all k executions of the DNN as independent and loses precision as a result of this. The SOTA DNN relational verifier (Zeng et al., 2023) (referred to as I/O formulation in the rest of the paper) although tracks the relationship between inputs used in multiple executions at the input layer, does not track the relationship between the inputs fed to the subsequent hidden layers and can only achieve a limited improvement over the baseline verifiers that treat all executions independently as shown in our experiments. There exist, probabilistic verifiers, (Xie et al., 2021; Zhang et al., 2022b) based on randomized smoothing (Cohen et al., 2019) for verifying relational properties. However, these works can only give probabilistic guarantees on smoothed models which have high inference costs. Similar to (Khedr & Shoukry, 2023; Zeng et al., 2023), in this work, we focus on deterministic scalable incomplete relational verifiers that can serve as a building block for BaB (Branch and Bound) based complete verifiers (Wang et al., 2021) with popular branching strategies like input splitting (Anderson et al., 2020), ReLU splitting (Wang et al., 2021), etc. We leave combining RACoon with branching strategies as future work. We consider DNNs with ReLU activation.

3. Preliminaries

We provide the necessary background on approaches for non-relational DNN verification, DNN safety properties that can be encoded as relational properties, and existing works on parametric bound refinement for individual executions.

Non-relational DNN verification: For individual execution, DNN verification involves proving that the network outputs $\mathbf{y} = N(\mathbf{x} + \boldsymbol{\delta})$ corresponding to all perturbations $\mathbf{x} + \boldsymbol{\delta}$ of an input $\mathbf{x}$ specified by ϕ, satisfy a logical specification ψ. For common safety properties like local DNN robustness, the output specification (ψ) is expressed as linear inequality (or conjunction of linear inequalities) over DNN output $\mathbf{y} \in \mathbb{R}^{n_l}$. e.g. $\psi(\mathbf{y}) = (\mathbf{c}^T\mathbf{y} \geq 0)$ where $\mathbf{c} \in \mathbb{R}^{n_l}$. In general, given a DNN $N : \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_l}$ and a property specified by (ϕ, ψ), scalable sound but incomplete verifiers compute a linear approximation specified by $\mathbf{L} \in \mathbb{R}^{n_0}, b \in \mathbb{R}$ such that for any input $\mathbf{x} \in \phi_t \subseteq \mathbb{R}^{n_0}$ satisfying ϕ the following condition holds $\mathbf{L}^T\mathbf{x} + b \leq \mathbf{c}^T N(\mathbf{x})$.

To show $\mathbf{c}^T N(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \phi_t$ DNN verifiers prove for all $\mathbf{x} \in \phi_t$, $\mathbf{L}^T\mathbf{x} + b \geq 0$ holds.

DNN relational properties: For a DNN $N : \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_l}$, relational properties defined over k executions of N are specified by the tuple (Φ, Ψ) where the input specification $\Phi : \mathbb{R}^{n_0 \times k} \rightarrow \{true, false\}$ encodes the input region $\Phi_t \subseteq \mathbb{R}^{n_0 \times k}$ encompassing all potential inputs corresponding to each of the k executions of N and the output specification $\Psi : \mathbb{R}^{n_l \times k} \rightarrow \{true, false\}$ specifies the safety property we expect the outputs of all k executions of N to satisfy. Formally, in DNN relational verification, given N, an input specification Φ and an output specification Ψ we require to prove whether $\forall \mathbf{x_1^*}, \ldots, \mathbf{x_k^*} \in \mathbb{R}^{n_0}.\Phi(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*}) \implies \Psi(N(\mathbf{x_1^*}), \ldots N(\mathbf{x_k^*}))$ or provide a counterexample otherwise. Here, $\mathbf{x_1^*}, \ldots, \mathbf{x_k^*}$ are the inputs to the k executions of N and $N(\mathbf{x_1^*}), \ldots, N(\mathbf{x_k^*})$ are the corresponding outputs. Commonly, the input region ϕ_t^i for the i-th execution is a L_∞ region around a fixed point $\mathbf{x_i} \in \mathbb{R}^{n_0}$ defined as $\phi_t^i = \{\mathbf{x_i^*} \in \mathbb{R}^{n_0} \mid \|\mathbf{x_i^*} - \mathbf{x_i}\|_\infty \leq \epsilon\}$ while the corresponding output specification $\psi^i(N(\mathbf{x_i^*})) = \bigwedge_{j=1}^{m}(\mathbf{c_{i,j}}^T N(\mathbf{x_i^*}) \geq 0)$. Subsequently, $\Phi(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*}) = \bigwedge_{i=1}^{k}(\mathbf{x_i^*} \in \phi_t^i) \bigwedge \Phi^\delta(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$ where $\Phi^\delta(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$ encodes the relationship between the inputs used in different execution and $\Psi(N(\mathbf{x_1^*}), \ldots, N(\mathbf{x_k^*})) = \bigwedge_{i=1}^{k} \psi^i(N(\mathbf{x_i^*}))$. Next, we describe interesting relational properties .

UAP verification: Given a DNN N, in a UAP attack, the adversary tries to find an adversarial perturbation with a bounded L_∞ norm that maximizes the misclassification rate of N when the same adversarial perturbation is applied to all inputs drawn from the input distribution. Conversely, the UAP verification problem finds the provably correct worst-case accuracy of N in the presence of a UAP adversary (referred to as UAP accuracy in the rest of the paper). (Zeng et al., 2023) showed that it is possible to statistically estimate (Theorem 2 in (Zeng et al., 2023)) UAP accuracy of N w.r.t input distribution provided we can characterize the UAP accuracy of N on k randomly selected images e.g. the k-UAP problem. For the rest of the paper, we focus on the k-UAP verification problem as improving the precision of k-UAP verification directly improves UAP accuracy on the input distribution (see Appendix E). The k-UAP verification problem fundamentally differs from the commonly considered local L_∞ robustness verification where the adversary can perturb each input independently. Since the adversarial perturbation is common across a set of inputs, the UAP verification problem requires a relational verifier that can exploit the dependency between perturbed inputs. We provide the input specification Φ and the output specification Ψ of the UAP verification problem in Appendix A.1.

Worst case hamming distance: The hamming distance between two strings with the same length is the number of substitutions needed to turn one string into the other (Hamming, 1950). Given a DNN N, a binary string (a list of images of binary digits), we want to formally verify the worst-case bounds on the hamming distance between the original binary string and binary string recognized by N where a common perturbation can perturb each image of the binary digits. Common perturbations are a natural consequence of faulty input devices that uniformly distort the inputs already considered in verification problems in (Paterson et al., 2021). The input specification Φ and the output specification Ψ are in Appendix A.2. Beyond hamming distance and k-UAP, RACoon is a general framework capable of formally analyzing the worst-case performance of algorithms that rely on multiple DNN executions (BANERJEE et al.). For example, the absolute difference between the original and the number recognized by a digit classifier.

Parametric bound refinement: Common DNN verifiers (Zhang et al., 2018; Singh et al., 2019b) handle non-linear activations $\sigma(x)$ in DNN by computing linear lower bound $\sigma_l(x)$ and upper bound $\sigma_u(x)$ that contain all possible outputs of the activation w.r.t the input region ϕ_t i.e. for all possible input values x, $\sigma_l(x) \leq \sigma(x) \leq \sigma_u(x)$ holds. Common DNN verifiers including the SOTA relational verifier (Zeng et al., 2023) also compute the linear bounds $\sigma_l(x)$ and $\sigma_u(x)$ statically without accounting for the property it is verifying. Recent works such as (Xu et al., 2021), instead of static linear bounds, use parametric linear bounds and refine the parameters with scalable differential optimization techniques to facilitate verification of the property (ϕ, ψ). For example, for $ReLU(x)$, the parametric lower bound is $ReLU(x) \geq \alpha \times x$ where the parameter $\alpha \in [0, 1]$ decides the slope of the lower bound. Since for any $\alpha \in [0, 1]$, $\alpha \times x$ is a valid lower bound of $ReLU(x)$ it is possible to optimize over α while ensuring mathematical correctness. Alternatively, (Salman et al., 2019) showed that optimizing α parameters is equivalent to optimizing the dual variables in the LP relaxed verification problem (Wong & Kolter, 2018). However, existing works can only optimize the α parameters w.r.t individual executions independently making these methods sub-optimal for relational verification. The key challenge here is to develop techniques for **jointly** optimizing α parameters over multiple DNN executions while leveraging their inter-dependencies.

4. Cross Executional Bound Refinement

Before delving into the details, first, we describe why it is essential to leverage cross-execution dependencies for relational verification. For illustrative purposes, we start with the k-UAP verification problem on a pair of executions i.e. $k = 2$. Note that bound refinement for worst-case hamming distance can be handled similarly. For 2-UAP, given a pair of unperturbed input $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{R}^{n_0}$ first we want to prove whether there exists an adversarial perturbation $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ with bounded L_∞ norm $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ such that N misclassifies both $(\mathbf{x_1} + \boldsymbol{\delta})$ and $(\mathbf{x_2} + \boldsymbol{\delta})$. Now, consider the scenario

where both $\mathbf{x_1}$ and $\mathbf{x_2}$ have valid adversarial perturbations $\boldsymbol{\delta_1}$ and $\boldsymbol{\delta_2}$ but no *common* perturbation say $\boldsymbol{\delta}$ that works for both $\mathbf{x_1}$ and $\mathbf{x_2}$. In this case, non-relational verification that does not account for cross-execution dependencies can never prove the absence of a common perturbation given that both $\mathbf{x_1}, \mathbf{x_2}$ have valid adversarial perturbations. This highlights the necessity of utilizing cross-execution dependencies. Next, we detail three key steps for computing a provably correct parametric linear approximation of N over multiple executions. So that the parameters from different executions are *jointly* optimized together to facilitate relational verification. Note that the SOTA relational verifier (Zeng et al., 2023) statically computes linear approximations of N independently without leveraging any dependencies.

LP formulation: Let, N correctly classify $(\mathbf{x_1} + \boldsymbol{\delta})$ if $\mathbf{c_1}^T N(\mathbf{x_1} + \boldsymbol{\delta}) \geq 0$ and $(\mathbf{x_2} + \boldsymbol{\delta})$ if $\mathbf{c_2}^T N(\mathbf{x_2} + \boldsymbol{\delta}) \geq 0$ where $\mathbf{c_1}, \mathbf{c_2} \in \mathbb{R}^{n_l}$. Then N does not have a common adversarial perturbation iff for all $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ the outputs $\mathbf{y_1} = N(\mathbf{x_1} + \boldsymbol{\delta})$ and $\mathbf{y_2} = N(\mathbf{x_2} + \boldsymbol{\delta})$ satisfy $\Psi(\mathbf{y_1}, \mathbf{y_2}) = (\mathbf{c_1}^T \mathbf{y_1} \geq 0) \vee (\mathbf{c_2}^T \mathbf{y_2} \geq 0)$. Any linear approximations specified with $\mathbf{L_1}, \mathbf{L_2} \in \mathbb{R}^{n_0}$ and $b_1, b_2 \in \mathbb{R}$ of N satisfying $\mathbf{L_1}^T(\mathbf{x_1} + \boldsymbol{\delta}) + b_1 \leq \mathbf{c_1}^T \mathbf{y_1}$ and $\mathbf{L_2}^T(\mathbf{x_2} + \boldsymbol{\delta}) + b_2 \leq \mathbf{c_2}^T \mathbf{y_2}$ for all $\boldsymbol{\delta}$ with $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ allow us to verify the absence of common adversarial perturbation with the following LP (linear programming) formulation.

$$min \;\; t \quad \text{s.t. } \|\boldsymbol{\delta}\|_\infty \leq \epsilon$$
$$\mathbf{L_1}^T(\mathbf{x_1} + \boldsymbol{\delta}) + b_1 \leq t, \mathbf{L_2}^T(\mathbf{x_2} + \boldsymbol{\delta}) + b_2 \leq t \quad (1)$$

Let t^* be the optimal solution of the LP formulation. Then $t^* \geq 0$ proves the absence of a common perturbation. For fixed linear approximations $\{(\mathbf{L_1}, b_1), (\mathbf{L_2}, b_2)\}$ of N, the LP formulation is exact i.e. it always proves the absence of common adversarial perturbation if it can be proved with $\{(\mathbf{L_1}, b_1), (\mathbf{L_2}, b_2)\}$ (see Theorem 4.1). This ensures that we do not lose any precision with the LP formulation and the LP formulation is more precise than any non-relational verifier using the same $\{(\mathbf{L_1}, b_1), (\mathbf{L_2}, b_2)\}$.

Theorem 4.1. $\vee_{i=1}^2 (\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \geq 0)$ *holds for all* $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ *with* $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ *if and only if* $t^* \geq 0$.
Proof: The proof follows from Appendix Theorem B.3.

However, the LP formulation only works with fixed $\{(\mathbf{L_1}, b_1), (\mathbf{L_2}, b_2)\}$ and as a result, is not suitable for handling parametric linear approximations that can then be optimized to improve the relational verifier's precision. Instead, we use the equivalent Lagrangian Dual (Boyd & Vandenberghe, 2004) which retains the benefits of the LP formulation while facilitating joint optimation of parameters from multiple executions as detailed below.

Dual with parametric linear approximations: Let, for a list of parametric activation bounds specified by a parameter list $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]$ we denote corresponding parametric linear approximation of N with the coefficient $\mathbf{L}(\boldsymbol{\alpha})$ and bias $\mathbf{b}(\boldsymbol{\alpha})$. First, for 2-UAP, we obtain $(\mathbf{L}_1(\boldsymbol{\alpha}_1), \mathbf{b}_1(\boldsymbol{\alpha}_1))$ and $(\mathbf{L}_2(\boldsymbol{\alpha}_2), \mathbf{b}_2(\boldsymbol{\alpha}_2))$ corresponding to the pair of executions using existing works (Xu et al., 2021). For $i \in \{1, 2\}$, $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ and $\mathbf{l}_i \preceq \boldsymbol{\alpha}_i \preceq \mathbf{u}_i$ the parametric linear bounds satisfy $\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i) \leq \mathbf{c_i}^T \mathbf{y_i}$ where $\mathbf{l}_i, \mathbf{u}_i$ are constant vectors defining valid range of the parameters $\boldsymbol{\alpha}_i$. For fixed $\boldsymbol{\alpha}_i$ the Lagrangian Dual of the LP formulation in Eq. 1 is as follows where $\lambda_1, \lambda_2 \in [0, 1]$ with $\lambda_1 + \lambda_2 = 1$ are the Lagrange multipliers relating linear approximations from different executions (details in Appendix B.1.2).

$$\max_{0 \leq \lambda_i \leq 1} \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum\nolimits_{i=1}^{2} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right)$$

Let, for fixed $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2$ the optimal solution of the dual formulation be $t^*(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$. Then we can prove the absence of common perturbation provided the maximum value of $t^*(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$ optimized over $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2$ is ≥ 0. This reduces the problem to the following: $\max t^*(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$ s.t. $\mathbf{l}_1 \preceq \boldsymbol{\alpha}_1 \preceq \mathbf{u}_1 \;\; \mathbf{l}_2 \preceq \boldsymbol{\alpha}_2 \preceq \mathbf{u}_2$. However, the optimization problem involves a max-min formulation and the number of parameters in $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2$ in the worst-case scales linearly with the number of activation nodes in N. This makes it hard to apply gradient descent-based techniques typically used for optimization (Xu et al., 2021). Instead, we reduce the max-min formulation to a simpler maximization problem by finding an optimizable closed form of the inner minimization problem.

Deriving optimizable closed form : We want to characterize the closed form $G(\boldsymbol{\lambda}) = \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{2} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right)$ where $\boldsymbol{\lambda} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \lambda_1, \lambda_2)$ and use it for formulating the maximization problem. Note, $G(\boldsymbol{\lambda})$ is related to the dual function from optimization literature (Boyd & Vandenberghe, 2004). Naively, it is possible to solve the inner minimization problem for two different executions separately and then optimize them over $\overline{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$ using $\overline{G}(\overline{\boldsymbol{\alpha}}) = max(\overline{G}_1(\boldsymbol{\alpha}_1), \overline{G}_2(\boldsymbol{\alpha}_2))$ as shown below. However, $\overline{G}(\overline{\boldsymbol{\alpha}})$ produces a suboptimal result since it ignores cross-execution dependencies and misses out on the benefits of jointly optimizing $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$.

$$\overline{G}_i(\boldsymbol{\alpha}_i) = \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i) \quad (2)$$

Since $\|\boldsymbol{\delta}\|_\infty$ is bounded by ϵ, it is possible to *exactly* compute the closed form of $G(\boldsymbol{\lambda})$ as shown below where for $j \in [n_0]$, $\mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \in \mathbb{R}$ denotes the j-th component of $\mathbf{L}_i(\boldsymbol{\alpha}_i) \in \mathbb{R}^{n_0}$ and $a_i(\boldsymbol{\alpha}_i) = \mathbf{L}_i(\boldsymbol{\alpha}_i)^T \mathbf{x_i} + \mathbf{b}_i(\boldsymbol{\alpha}_i)$

$$G(\boldsymbol{\lambda}) = \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{2} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right)$$

$$G(\boldsymbol{\lambda}) = \sum_{i=1}^{2} \lambda_i \times a_i(\boldsymbol{\alpha}_i) + \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{2} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T \boldsymbol{\delta}$$

$$G(\boldsymbol{\lambda}) = \sum_{i=1}^{2} \lambda_i \times a_i(\boldsymbol{\alpha}_i) - \epsilon \times \sum_{j=1}^{n_0} \left| \sum_{i=1}^{2} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right|$$

Unlike $\overline{G}(\overline{\boldsymbol{\alpha}})$, $G(\boldsymbol{\lambda})$ relates linear approximations from two different executions using (λ_1, λ_2) enabling joint optimization over $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$. With the closed form $G(\boldsymbol{\lambda})$, we can use projected gradient descent to optimize $max_{\boldsymbol{\lambda}} G(\boldsymbol{\lambda})$ while ensuring the parameters in $\boldsymbol{\lambda}$ satisfy the corresponding constraints. Next, we provide theoretical guarantees about the correctness and efficacy of the proposed technique. For efficacy, we show the optimal solution $t^*(G)$ obtained with $G(\boldsymbol{\lambda})$ is always as good as $t^*(\overline{G})$ i.e. $t^*(G) \geq t^*(\overline{G})$ (Theorem 4.2) and characterize sufficient condition where $t^*(G)$ is strictly better i.e. $t^*(G) > t^*(\overline{G})$ (Appendix Theorem B.7). Experiments substantiating the improvement in the optimal values ($t^*(G)$ vs. $t^*(\overline{G})$) are in Section 6.2.

Theorem 4.2. *If* $t^*(G) = \max_{\boldsymbol{\lambda}} G(\boldsymbol{\lambda})$ *and* $t^*(\overline{G}) = \max_{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2} \overline{G}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$ *then* $t^*(\overline{G}) \leq t^*(G)$.

Proof: For any $\mathbf{l}_1 \preceq \boldsymbol{\alpha}_1 \preceq \mathbf{u}_1 \;\; \mathbf{l}_2 \preceq \boldsymbol{\alpha}_2 \preceq \mathbf{u}_2$, consider $\boldsymbol{\lambda}_1 = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \lambda_1 = 1, \lambda_2 = 0)$ and $\boldsymbol{\lambda}_2 = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \lambda_1 = 0, \lambda_2 = 1)$, then $G(\boldsymbol{\lambda}_1) = \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \mathbf{L}_1(\boldsymbol{\alpha}_1)^T(\mathbf{x_1} + \boldsymbol{\delta}) + \mathbf{b}_1(\boldsymbol{\alpha}_1)$ and $G(\boldsymbol{\lambda}_2) = \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \mathbf{L}_2(\boldsymbol{\alpha}_2)^T(\mathbf{x_2} + \boldsymbol{\delta}) + \mathbf{b}_2(\boldsymbol{\alpha}_2)$. Since, $t^*(G) \geq G(\boldsymbol{\lambda}_1)$ and $t^*(G) \geq G(\boldsymbol{\lambda}_2)$ then $t^*(G) \geq \max_{1 \leq i \leq 2} G(\boldsymbol{\lambda}_i) = \overline{G}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$. Hence, $t^*(G) \geq \max_{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2} \overline{G}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2) = t^*(\overline{G})$.

The correctness proof for bound refinement between two executions is in Appendix B.1.3. Note that correctness does not necessitate the optimization technique to identify the global maximum, especially since gradient-descent-based optimizers may not always find the global maximum. The details of the optimization step is in Appendix B.4.

Genralization to multiple executions: Instead of a pair of executions considered above, we now generalize the approach to any set of n executions where $n \leq k$. With parametric linear approximations $\{(\mathbf{L}_1, b_1), \ldots, (\mathbf{L}_n, b_n)\}$ of N for all n executions, we formulate the following LP to prove the absence of common adversarial perturbation that works for *all* n executions. The proof of exactness of the LP formulation is in Appnedix Theorem B.3.

$$\begin{aligned} &min \;\; t \quad \text{s.t. } \|\boldsymbol{\delta}\|_\infty \leq \epsilon \\ &\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \leq t \quad \forall i \in [n] \end{aligned} \tag{3}$$

Similar to a pair of executions, we first specify the Lagrangian dual of the LP (Eq. 3) by introducing n lagrangian multipliers $\lambda_1, \ldots, \lambda_n$ that satisfy for all $i \in [n]$ $\lambda_i \in [0, 1]$ and $\sum_{i=1}^n \lambda_i = 1$. Subsequently, we obtain the closed form $G(\boldsymbol{\lambda})$ where $\boldsymbol{\lambda} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n, \lambda_1, \ldots, \lambda_n)$ and $a_i(\boldsymbol{\alpha}_i) = \mathbf{L}_i(\boldsymbol{\alpha}_i)^T\mathbf{x_i} + \mathbf{b}_i(\boldsymbol{\alpha}_i)$ as shown below.

$$G(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) - \epsilon \times \sum_{j=1}^{n_0} \left| \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right|$$

Theoretical results regarding the correctness and efficacy of bound computation over n executions are in Appendix B.1.

Genralization to a conjunction of linear inequalities: Until now, we assume for each execution the output specification is defined as a linear inequality i.e. $\mathbf{c_i}^T N(\mathbf{x_i} + \boldsymbol{\delta}) \geq 0$. Next, we generalize our method to any output specification for each execution defined with conjunction of m linear inequalities. For example, if $\mathbf{y_i}$ denotes the output of the i-th execution $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$ then the output specification $\psi^i(\mathbf{y_i})$ is given by $\psi^i(\mathbf{y_i}) = \bigwedge_{j=1}^m (\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$ where $\mathbf{c_{i,j}} \in \mathbb{R}^{n_l}$. In this case, $\psi(\mathbf{y_i})$ is satisfied iff $(\min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i}) \geq 0$. Using this observation, we first reduce this problem to subproblems with a single linear inequality (see Appendix Theorem B.10) and subsequently characterize the closed form $G(\boldsymbol{\lambda})$ for each subproblem separately. However, the number of subproblems in the worst case can be m^n which is practically intractable for large m and n. Hence, we greedily select which subproblems to use for bound refinement to avoid exponential blow-up in the runtime while ensuring the bound refinement remains provably correct (see Appendix B.2.1). Since most of the common DNN output specification can be expressed as a conjunction of linear inequalities (Zhang et al., 2018) RACoon generalizes to them. Moreover, cross-excution bound refinement is not restricted to L_∞ input specification where $\|\boldsymbol{\delta}\|_\infty$ is bounded and can work for any $\|\cdot\|_p$ norm bounded perturbation (see Appendix B.3).

Next, we utilize the cross-executional bound refinement to formulate a MILP with at most $O(k \times n_l)$ integer variables. Similar to (Zeng et al., 2023) we only use integer variables to encode the output specification Ψ. Since the output dimension n_l of N is usually much smaller than the number of total ReLU nodes $n_l << n_r$ in N, RACoon is more scalable than the naive MILP encoding that in the worst case introduces $O(k \times n_r)$ integer variables.

5. RACoon Algorithm

The cross-executional bound refinement learns parameters over any set of n executions. However, for a relational property defined over k executions, since there are $2^k - 1$ non-empty subsets of executions, refining bounds for all possible subsets is impractical. Instead, we design a greedy heuristic to pick the subsets of executions so that we only use a small number of subsets for bound refinement.

Eliminating individually verified executions: First, we run existing non-relational verifiers (Zhang et al., 2018; Singh et al., 2019b) without tracking any dependencies across executions. RACoon eliminates the executions already verified with the non-relational verifier and does not consider them for subsequent steps. (lines 5 – 9 in Algo. 1) For example, for the k-UAP property, we do not need to consider those executions that are proved to have no adversarial perturbation $\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$. For relational properties considered in this paper, we formally prove the correctness of the elimination technique in Appendix Theorem B.12 and

Algorithm 1 RACoon

1: **Input:** N, (Φ, Ψ), k, k_0, k_1, non-relational verifier $\mathcal{V}$.
2: **Output:** sound approximation of worst-case k-UAP accuracy or worst-case hamming distance $\mathbf{M}(\Phi, \Psi)$.
3: $I \leftarrow \{\}$. {Indices of executions not verified by $\mathcal{V}$}
4: $\mathcal{L} \leftarrow \{\}$ {Map storing linear approximations}
5: **for** $i \in [k]$ **do**
6: $(s_i, \mathbf{L}_i, b_i) \leftarrow \mathcal{V}(\phi^i, \psi^i)$.
7: **if** $\mathcal{V}$ can not verify (ϕ^i, ψ^i) **then**
8: $I \leftarrow I \cup \{i\}$; $\mathcal{L}[i] \leftarrow \mathcal{L}[i] \cup (\mathbf{L}_i, b_i)$.
9: **end if**
10: **end for**
11: $I_0 \leftarrow$ top-k_0 executions from I selected based on s_i.
12: **for** $\overline{I_0} \subseteq I_0$, $\overline{I_0} \neq \{\}$ and $|\overline{I_0}| \leq k_1$ **do**
13: $\mathcal{L}_{\overline{I_0}} \leftarrow$ CrossExcutionalRefinement$(\overline{I_0}, \Phi, \Psi)$.
14: $\mathcal{L} \leftarrow$ Populate$(\mathcal{L}, \mathcal{L}_{\overline{I_0}})$. {Storing $\mathcal{L}_{\overline{I_0}}$ in $\mathcal{L}$}
15: **end for**
16: $\mathcal{M} \leftarrow$MILPFormulation$(\mathcal{L}, \Phi, \Psi, k, I)$.
17: **return** Optimize$(\mathcal{M})$.

showcase eliminating verified executions does not lead to any loss in precision of RACoon.

Greedy selection of unverified executions: For each execution that remains unverified with the non-relational verifier ($\mathcal{V}$), we look at $s_i = \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i}$ estimated by $\mathcal{V}$ where $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$ and $\mathbf{c_{i,j}} \in \mathbb{R}^{n_l}$ defines the corresponding output specification $\psi^i(\mathbf{y_i}) = \bigwedge_{i=1}^{m} (\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$. Intuitively, for unverified executions, s_i measures the maximum violation of the output specification $\psi^i(\mathbf{y_i})$ and thus leads to the natural choice of picking executions with smaller violations for cross-executional refinement. We sort the executions in decreasing order of s_i and pick the first k_0 (hyperparameter) executions on input regions $\mathbf{X} = \{\phi_t^1, \ldots, \phi_t^{k_0}\}$ having smaller violations s_i where for all $i \in [k_0]$, $\phi_t^i = \{\mathbf{x'_i} + \boldsymbol{\delta} \mid \mathbf{x'_i},\ \boldsymbol{\delta} \in \mathbb{R}^{n_0} \wedge \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$ and $\mathbf{x'_i}$ is the unperturbed input. (line 11 of Algo. 1) In general, k_0 is a small constant i.e. $k_0 \leq 10$. Further, we limit the subset size to k_1 (hyperparameter) and do not consider any subset of $\mathbf{X}$ with a size more than k_1 for cross-executional bound refinement. (lines 12 – 15 in Algo. 1) Overall, we consider $\sum_{i=1}^{k_1} \binom{k_0}{i}$ subsets for bound refinement.

MILP formulation: RACoon MILP formulation involves two steps. First, we deduce linear constraints between the input and output of N for each unverified execution using linear approximations of N either obtained through cross-executional refinement or by applying the non-relational verifier. Secondly, similar to the current SOTA baseline (Zeng et al., 2023) we encode the output specification Ψ as MILP objective that only introduces $O(k \times n_l)$ integer variables. Finally, we use an off-the-shelf MILP solver (Gurobi Optimization, LLC, 2018) to optimize the MILP.

For the i-th unverified execution, let $\phi_t^i = \{\mathbf{x'_i} + \boldsymbol{\delta} \mid \mathbf{x'_i},\ \boldsymbol{\delta} \in \mathbb{R}^{n_0} \wedge \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$ be the input region and for $\mathbf{y_i} = N(\mathbf{x'_i} + \boldsymbol{\delta})$, $\psi^i(\mathbf{y_i}) = \bigwedge_{i=1}^{m} (\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$ be the output specification. Subsequently for each clause $(\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$ in $\psi^i(\mathbf{y_i})$ let $\{(\mathbf{L}_{i,j}^1, b_{i,j}^1), \ldots, (\mathbf{L}_{i,j}^{k'}, b_{i,j}^{k'})\}$ be set of linear approximations. Then for each $l \in [k']$ we add the following linear constraints where $o_{i,j}$ is a real variable.

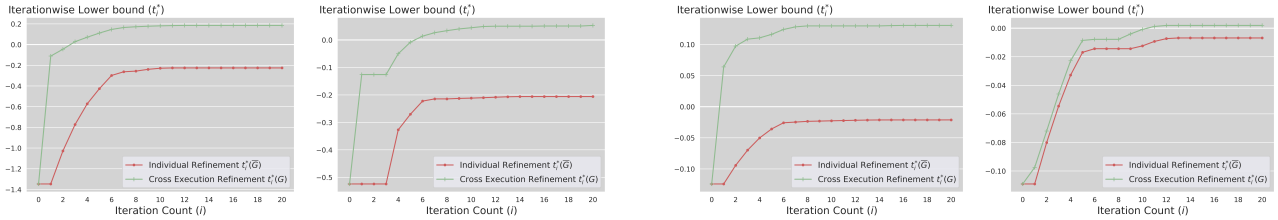
$$\mathbf{L}_{i,j}^l(\mathbf{x'_i} + \boldsymbol{\delta}) + b_{i,j}^l \leq o_{i,j}\ ;\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon$$

Next, similar to (Zeng et al., 2023) we encode output specification (ψ^i) as $z_i = (\min_{1 \leq j \leq m} o_{i,j}) \geq 0$ where $z_i \in \{0, 1\}$ are binary variables and $z_i = 1$ implies $\psi^i(y_i) = True$. Encoding of each ψ^i introduces $O(m)$ binary (integer) variables. Since for k-UAP and worst-case hamming distance, $m = n_l$ the total number of integer variables is in the worst case $O(k \times n_l)$. MILP encoding for k-UAP and worst-case hamming distance verification are shown in Appendix B.5.1. We prove the correctness of RACoon in Appendix Theorem B.13 and show it is always at least as precise as (Zeng et al., 2023) (Appendix Theorem B.14). Worst-case time complexity analysis of RACoon is in Appendix C.

Limitation: Similar to other deterministic (relational or non-relational) verifiers RACoon does not scale to DNNs trained on larger datasets (e.g. ImageNet). RACoon is sound but incomplete and for some cases, RACoon may fail to prove a property even if the property holds. However, for piecewise linear activations like ReLU, it is possible to design a "Branch and Bound" based complete relational verifiers by combining RACoon (as bounding algorithm) with branching algorithms like ReLU splitting (Wang et al., 2021). We leave that as future work. Note that existing complete non-relational verifiers like (Wang et al., 2021) are incomplete for relational properties since they can only verify each execution in isolation.

6. Experimental Evaluation

We evaluate the effectiveness of RACoon on a wide range of relational properties and a diverse set of DNNs and datasets. We consider the following relational properties: k-UAP, worst-case hamming distance as formally defined in Appendix A. The baselines we consider are the SOTA relational verifier (Zeng et al., 2023) (referred to as I/O Formulation) and the non-relational verifier (Xu et al., 2020) from the SOTA auto_LiRPA toolbox (Xu et al., 2020). used by (Zeng et al., 2023). We also analyze the efficacy of cross-executional bound refinement in learning parametric bounds that can facilitate relational verification (Section 6.2). Note that we instantiate RACoon with the same non-relational verifier (Xu et al., 2020) used in I/O formulation (Zeng et al., 2023). The performance evaluation of different components of RACoon including individual bound refinement (i.e. refinement on execution set of size 1), and individual bound refinement with MILP is in Appendix Table 4. Note that in-



. (a) MNIST (PGD) . (b) MNIST (DiffAI) . (c) CIFAR10 (PGD) . (d) CIFAR10 (DiffAI)

Figure 1. Lower bound (t in Eq. 3) from individual vs. cross executional bound refinement over 2 executions on ConvSmall networks.

dividual bound refinement uses SOTA non-relational bound refinement algorithm α-CROWN (Xu et al., 2021).

6.1. Experiment setup

Networks. We use standard convolutional architectures (ConvSmall, ConvBig, etc.) commonly seen in other neural network verification works (Zhang et al., 2018; Singh et al., 2019b) (see Table 1). Details of DNN architectures used in experiments are in Appendix D. We consider networks trained with standard training, robust training: DiffAI (Mirman et al., 2018), CROWN-IBP (Zhang et al., 2020), projected gradient descent (PGD) (Madry et al., 2018), and COLT (Balunovic & Vechev, 2020). We use pre-trained publically available DNNs: CROWN-IBP DNNs taken from the CROWN repository (Zhang et al., 2020) and all other DNNs are from the ERAN repository (Singh et al., 2019b).

Implementation Details. The details regarding the frameworks RACoon uses, the CPU and GPU information, and the hyperparameter (k_0, k_1) values are in Appendix D.1.

6.2. Evaluating cross execution bound refinement

Fig. 1 shows the values $t_i^*(G)$ and $t_i^*(\overline{G})$ after i-th iteration of Adam optimizer computed by cross-executional and individual refinement (using α-CROWN) respectively over a pair of executions (i.e. $k = 2$) on randomly chosen images. We used ConvSmall PGD and DiffAI DNNs trained on MNIST and CIFAR10 for this experiment. The ϵs used for MNIST PGD and DiffAI DNNs are 0.1 and 0.12 respectively while ϵs used for CIFAR10 PGD and DiffAI DNNs are $2.0/255$ and $6.0/255$ respectively. For each iteration i, $t_i^*(G) > t_i^*(\overline{G})$ shows that cross-executional refinement is more effective in learning parametric bounds that can facilitate relation verification. Since, for proving the absence of common adversarial perturbation, we need to show $t^* \geq 0$, in all 4 cases in Fig. 1 individual refinement fails to prove the absence of common adversarial perturbation while cross-executional refinement succeeds. Moreover, in all 4 cases, even the optimal solution of the LP (Eq. 3) formulated with linear approximations from individual refinement remains negative. For example, for MNIST DiffAI DNN, with LP, $t^*(\overline{G})$ improves to -0.05 from -0.2 but remains insufficient for proving the absence of common adversarial perturbation. This shows the importance of leveraging dependencies across executions during bound refinement.

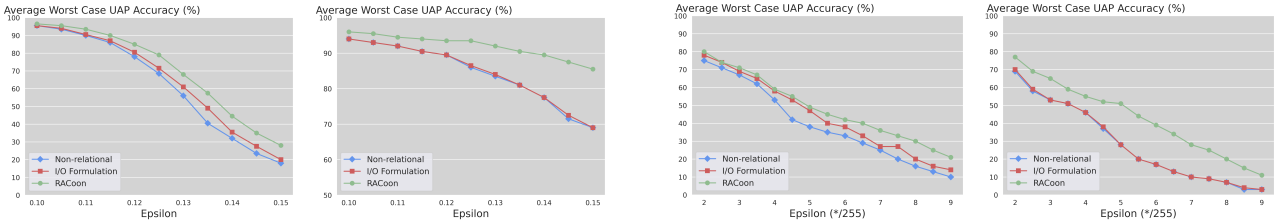
Verification results: For k-UAP, both the baselines: non-relational verifier (Xu et al., 2020), I/O formulation (Zeng et al., 2023) and RACoon computes a provably correct lower bound $\mathbf{M}(\Phi, \Psi)$ on the worst-case UAP accuracy. In this case, larger $\mathbf{M}(\Phi, \Psi)$ values produce a more precise lower bound tightly approximating the actual worst-case UAP accuracy. In contrast, for worst-case hamming distance $\mathbf{M}(\Phi, \Psi)$ is a provably correct upper bound and smaller $\mathbf{M}(\Phi, \Psi)$ values are more precise. Table 1 shows the verification results on different datasets (column 1), DNN architectures (column 3) trained with different training methods (column 4) where ϵ values defining L_∞ bound of $\boldsymbol{\delta}$ are in column 5. The relational properties: k-UAP and worst-case hamming distance on MNIST DNNs use $k = 20$ while k-UAP on CIFAR10 DNNs uses $k = 10$. For each DNN and ϵ, we run relational verification on k randomly selected inputs and repeat the experiment 10 times. We report worst-case UAP accuracy and worst-case hamming distance averaged over all 10 runs. Results in Table 1 substantiate that RACoon outperforms current SOTA baseline I/O formulation on all DNNs for both the relational properties. RACoon gains up to $+16.5\%$ and up to $+22\%$ improvement in the worst-case UAP accuracy (averaged over 10 runs) for MNIST and CIFAR10 DNNs respectively. Similarly, RACoon reduces the worst-case hamming distance (averaged over 10 runs) up to 8 which is up to 40% reduction for binary strings of size 20.

Runtime analysis: Table 1 shows that RACoon is slower than I/O formulation. However, even for ConvBig architectures, RACoon takes less than 8 seconds (for 20 executions) for MNIST and takes less than 12 seconds (for 10 executions) for CIFAR10. The timings are much smaller compared to the timeouts allotted for similar architectures in the SOTA competition for verification of DNNs (VNN-Comp (Brix et al., 2023)) (200 seconds per execution).

RACoon componentwise analysis: In Appendix Table 4, we show results for different components of RACoon including individual bound refinement using $\overline{G}$ (Eq. 2), individual bound refinement with MILP formulation, and cross-executional bound refinement without MILP formulation.

Table 1. RACoon Efficacy Analysis

Dataset	Property	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier Avg. UAP Acc. (%)	Non-relational Verifier Avg. Time (sec.)	I/O Formulation Avg. UAP Acc. (%)	I/O Formulation Avg. Time (sec.)	RACoon Avg. UAP Acc. (%)	RACoon Avg. Time (sec.)
MNIST	UAP	ConvSmall	Standard	0.08	38.5	0.01	48.0	2.65	54.0 (+6.0)	5.20
	UAP	ConvSmall	PGD	0.10	70.5	0.21	72.0	0.92	77.0 (+5.0)	4.33
	UAP	IBPSmall	IBP	0.13	74.5	0.02	75.0	1.01	89.0 (+14.0)	2.01
	UAP	ConvSmall	DiffAI	0.13	56.0	0.01	61.0	1.10	68.0 (+7.0)	3.98
	UAP	ConvSmall	COLT	0.15	69.0	0.02	69.0	0.99	85.5 (+16.5)	2.68
	UAP	IBPMedium	IBP	0.20	80.5	0.1	82.0	0.99	93.5 (+11.5)	2.30
	UAP	ConvBig	DiffAI	0.20	81.5	1.85	81.5	2.23	91.5 (+10.0)	7.60
CIFAR10	UAP	ConvSmall	Standard	1.0/255	52.0	0.02	55.0	3.46	58.0 (+3.0)	7.22
	UAP	ConvSmall	PGD	3.0/255	21.0	0.01	26.0	1.57	29.0 (+3.0)	5.56
	UAP	IBPSmall	IBP	6.0/255	17.0	0.02	17.0	2.76	39.0 (+22.0)	6.76
	UAP	ConvSmall	DiffAI	8.0/255	16.0	0.01	20.0	2.49	30.0 (+10.0)	7.09
	UAP	ConvSmall	COLT	8.0/255	18.0	0.04	21.0	2.41	26.0 (+5.0)	11.02
	UAP	IBPMedium	IBP	3.0/255	46.0	0.15	50.0	2.13	71.0 (+21.0)	6.12
	UAP	ConvBig	DiffAI	3.0/255	17.0	1.33	20.0	3.42	25.0 (+5.0)	11.92
Dataset	Property	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier Avg. Hamming distance	Non-relational Verifier Avg. Time (sec.)	I/O Formulation Avg. Hamming distance	I/O Formulation Avg. Time (sec.)	RACoon Avg. Hamming distance	RACoon Avg. Time (sec.)
MNIST	Hamming	ConvSmall	Standard	0.10	19.0	0.01	18.0	2.68	16.0 (-2.0)	4.43
	Hamming	ConvSmall	PGD	0.12	17.0	0.01	16.0	0.99	14.0 (-2.0)	3.20
	Hamming	ConvSmall	DiffAI	0.15	16.0	0.01	16.0	0.98	14.0 (-2.0)	3.46
	Hamming	IBPSmall	IBP	0.14	11.0	0.01	10.0	1.13	5.0 (-5.0)	2.56
	Hamming	ConvSmall	COLT	0.20	17.0	0.01	17.0	0.89	10.0 (-7.0)	1.88
	Hamming	IBPMedium	IBP	0.30	12.0	0.02	11.0	0.87	3.0 (-8.0)	1.75



. (a) DiffAI (MNIST) . (b) IBPSmall (MNIST) . (c) DiffAI (CIFAR10) . (d) IBPSmall (CIFAR10)

Figure 2. Average Worst case UAP accuracy for different ϵ values for ConvSmall (DiffAI) and IBPSmall DNNs.

Note that only cross-executional bound refinement without MILP can prove the absence of common adversarial perturbation for a set of executions even if non-relational verification fails on all of them. Hence, even without MILP, cross-executional bound refinement serves as a promising approach for relational verification. Appendix Table 4 shows for some cases (i.e. MNIST and CIFAR10 standard DNNs) I/O formulation (static linear approximation with MILP) outperforms individual refinements while both individual refinement with MILP and cross-execution refinement are always more precise. As expected, RACoon (cross-execution refinement with MILP) yields the most precise results while cross-execution refinement without MILP achieves the second-best results with notably faster runtime. Componentwise runtime analysis is in Appendix G.

Different ϵ and k values: Fig. 2 and Appendix Fig. 3, 4 show the results of RACoon and both the baselines on relational properties defined with different ϵ values on DNNs from Table 1. We also analyze the performance of RACoon for k-UAP verification defined with different k and ϵ values in Appendix I on DNNs from Table 1. For the MNIST DNNs, we consider up to 50 executions, and for CIFAR10 DNNs we consider up to 25 executions per property. For all k and ϵ values RACoon is more precise than both baselines. In all cases, even for ConvBig MNIST and CIFAR10, RACoon takes less than 16 and 25 seconds respectively.

Ablation on hyperparameters k_0 and k_1: We analyze the impact of k_0 and k_1 on performance of RACoon in Appendix J. As expected, with larger k_0 and k_1 RACoon's precision improves but it also increases RACoon's runtime.

7. Conclusion

In this work, we present RACoon, a general framework for improving the precision of relational verification of DNNs through cross-executional bound refinement. Our experiments, spanning various relational properties, DNN architectures, and training methods demonstrate the effectiveness of utilizing dependencies across multiple executions. Furthermore, RACoon with cross-executional bound refinement proves to exceed the capabilities of the current state-of-the-art relational verifier (Zeng et al., 2023). While our focus has been on relational properties within the same DNN across multiple executions, RACoon can be extended to properties involving different DNNs, such as local equivalence of DNN pairs (Paulsen et al., 2020) or properties defined over an ensemble of DNNs. Additionally, RACoon can be leveraged for training DNNs on relational properties. We leave these extensions as future work.

Impact Statement

This paper introduces research aimed at advancing the field of Machine Learning. We do not identify any specific societal consequences of our work that need to be explicitly emphasized here.

Acknowledgement

We thank the anonymous reviewers for their comments. This research was supported in part by NSF Grants No. CCF-2238079, CCF-2316233, CNS-2148583, and Google Research Scholar award.

References

Amato, F., López, A., Peña-Méndez, E. M., Vaňhara, P., Hampl, A., and Havel, J. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, 11 (2), 2013.

Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., and Vielma, J. P. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 2020.

Bak, S., Tran, H., Hobbs, K., and Johnson, T. T. Improved geometric path enumeration for verifying relu neural networks. In Lahiri, S. K. and Wang, C. (eds.), *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, pp. 66–96. Springer, 2020. doi: 10.1007/978-3-030-53288-8_4. URL https://doi.org/10.1007/978-3-030-53288-8_4.

Balunovic, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxSDxrKDr.

BANERJEE, D., XU, C., and SINGH, G. Input-relational verification of deep neural networks.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brix, C., Müller, M. N., Bak, S., Johnson, T. T., and Liu, C. First three years of the international verification of neural networks competition (vnn-comp). *International Journal on Software Tools for Technology Transfer*, pp. 1–11, 2023.

Bunel, R., Lu, J., Turkaslan, I., Kohli, P., Torr, P., and Mudigonda, P. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(2020), 2020a.

Bunel, R. R., Hinder, O., Bhojanapalli, S., and Dvijotham, K. An efficient nonconvex reformulation of stagewise convex optimization problems. *Advances in Neural Information Processing Systems*, 33, 2020b.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1310–1320. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/cohen19c.html.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 2017.

Ferrari, C., Mueller, M. N., Jovanović, N., and Vechev, M. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=l_amHf1oaK.

Fromherz, A., Leino, K., Fredrikson, M., Parno, B., and Pasareanu, C. Fast geometric projections for local robustness certification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zWy1uxjDdZJ.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2018.

Hamming, R. W. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

Khedr, H. and Shoukry, Y. Certifair: A framework for certified global fairness of neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7): 8237–8245, Jun. 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Li, J., Qu, S., Li, X., Szurley, J., Kolter, J. Z., and Metze, F. Adversarial music: Real world audio adversary against wake-word detection system. In *Proc. Neural Information Processing Systems (NeurIPS)*, pp. 11908–11918, 2019a.

Li, J., Schmidt, F. R., and Kolter, J. Z. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *Proc. International Conference on Machine Learning, ICML*, volume 97, pp. 3896–3904, 2019b.

Li, L., Zhang, J., Xie, T., and Li, B. Double sampling randomized smoothing. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13163–13208. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22aa.html.

Liu, Z., Xu, C., Sie, E., Singh, G., and Vasisht, D. Exploring practical vulnerabilities of machine learning-based wireless systems. In *20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023*, pp. 1801–1817. USENIX Association, 2023.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3578–3586. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/mirman18b.html.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

Palma, A. D., Behl, H. S., Bunel, R. R., Torr, P. H. S., and Kumar, M. P. Scaling the convex barrier with active sets. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Paterson, C., Wu, H., Grese, J., Calinescu, R., Păsăreanu, C. S., and Barrett, C. Deepcert: Verification of contextually relevant robustness for neural network image classifiers. In Habli, I., Sujan, M., and Bitsch, F. (eds.), *Computer Safety, Reliability, and Security*, pp. 3–17, Cham, 2021. Springer International Publishing. ISBN 978-3-030-83903-1.

Paulsen, B., Wang, J., and Wang, C. Reludiff: Differential verification of deep neural networks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, pp. 714–726, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371216. doi: 10.1145/3377811.3380337. URL https://doi.org/10.1145/3377811.3380337.

Paulsen, B., Wang, J., Wang, J., and Wang, C. Neurodiff: Scalable differential verification of neural networks using fine-grained approximation. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, pp. 784–796, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450367684. doi: 10.1145/3324884.3416560. URL https://doi.org/10.1145/3324884.3416560.

Potdevin, Y., Nowotka, D., and Ganesh, V. An empirical investigation of randomized defenses against adversarial attacks. *arXiv preprint arXiv:1909.05580*, 2019.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/246a3c5544feb054f3ea718f61adfa16-Paper.pdf.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 31, 2018.

Singh, G., Ganvir, R., Püschel, M., and Vechev, M. Beyond the single neuron convex barrier for neural network certification. In *Advances in Neural Information Processing Systems*, 2019a.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL), 2019b.

Sotoudeh, M. and Thakur, A. V. Abstract neural networks. In *Static Analysis: 27th International Symposium, SAS*

2020, Virtual Event, November 18–20, 2020, Proceedings 27, pp. 65–88. Springer, 2020.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5283–5292. PMLR, 2018. URL http://proceedings.mlr.press/v80/wong18a.html.

Wu, H., Tagomori, T., Robey, A., Yang, F., Matni, N., Pappas, G., Hassani, H., Pasareanu, C., and Barrett, C. Toward certified robustness against real-world distribution shifts. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 537–553. IEEE, 2023.

Xie, C., Chen, M., Chen, P.-Y., and Li, B. Crfl: Certifiably robust federated learning against backdoor attacks. In *International Conference on Machine Learning*, pp. 11372–11382. PMLR, 2021.

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., and Hsieh, C.-J. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nVZtXBI6LNn.

Zeng, Y., Shi, Z., Jin, M., Kang, F., Lyu, L., Hsieh, C.-J., and Jia, R. Towards robustness certification against universal perturbations. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=7GEvPKxjtt.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *Proc. International Conference on Learning Representations (ICLR)*, 2020.

Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hsieh, C.-J., and Kolter, J. Z. General cutting planes for bound-propagation-based neural network verification. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=5haAJAcofjc.

Zhang, Y., Albarghouthi, A., and D'Antoni, L. Bagflip: A certified defense against data poisoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=ZidkM5b92G.

A. Formal encoding of relational properties

A.1. k-UAP verification

Given a set of k points $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_k}\}$ where for all $i \in [k]$, $\mathbf{x_i} \in \mathbb{R}^{n_0}$ and $\epsilon \in \mathbb{R}$ we can first define individual input constraints used to define L_∞ input region for each execution $\forall i \in [k].\phi^i_{in}(\mathbf{x_i^*}) = \|\mathbf{x_i^*} - \mathbf{x_i}\|_\infty \leq \epsilon$. We define $\Phi^\delta(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$ as follows:

$$\Phi^\delta(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*}) = \bigwedge_{(i,j \in [k]) \wedge (i<j)} (\mathbf{x_i^*} - \mathbf{x_j^*} = \mathbf{x_i} - \mathbf{x_j}) \tag{4}$$

Then, we have the input specification as $\Phi(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*}) = \bigwedge_{i=1}^{k} \phi^i_{in}(\mathbf{x_i^*}) \wedge \Phi^\delta(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$.

Next, we define $\Psi(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$ as conjunction of k clauses each defined by $\psi^i(\mathbf{y_i})$ where $\mathbf{y_i} = N(\mathbf{x_i^*})$. Now we define $\psi^i(\mathbf{y_i}) = \bigwedge_{j=1}^{n_l} (\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$ where $\mathbf{c_{i,j}} \in \mathbb{R}^{n_l}$ is defined as follows

$$\forall a \in [n_l].c_{i,j,a} = \begin{cases} 1 & \text{if } a \neq j \text{ and } a \text{ is the correct label for } \mathbf{y_i} \\ -1 & \text{if } a = j \text{ and } a \text{ is not the correct label for } \mathbf{y_i} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In this case, the tuple of inputs $(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$ satisfies the input specification $\Phi(\mathbf{x_1^*}, \ldots, \mathbf{x_k^*})$ iff for all $i \in [k]$, $\mathbf{x_i^*} = \mathbf{x_i} + \boldsymbol{\delta}$ where $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$. Hence, the relational property (Φ, Ψ) defined above verifies whether there is an adversarial perturbation $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ with $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ that can misclassify **all** k inputs. Next, we show the formulation for the worst-case UAP accuracy of the k-UAP verification problem as described in section 3. Let, for any $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, $\mu(\delta)$ denotes the number of clauses (ψ^i) in Ψ that are satisfied. Then $\mu(\delta)$ is defined as follows

$$z_i(\boldsymbol{\delta}) = \begin{cases} 1 & \psi^i(N(\mathbf{x_i} + \boldsymbol{\delta})) \text{ is } True \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\mu(\boldsymbol{\delta}) = \sum_{i=1}^{k} z_i(\boldsymbol{\delta}) \tag{7}$$

Since $\psi^i(N(\mathbf{x_i} + \boldsymbol{\delta}))$ is $True$ iff the perturbed input $\mathbf{x_i} + \boldsymbol{\delta}$ is correctly classified by N, for any $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, $\mu(\boldsymbol{\delta})$ captures the number of correct classifications over the set of perturbed inputs $\{\mathbf{x_1} + \boldsymbol{\delta}, \ldots, \mathbf{x_k} + \boldsymbol{\delta}\}$. The worst-case k-UAP accuracy $\mathbf{M}_0(\Phi, \Psi)$ for (Φ, Ψ) is as follows

$$\mathbf{M}_0(\Phi, \Psi) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\| \leq \epsilon} \mu(\boldsymbol{\delta}) \tag{8}$$

A.2. Worst case Hamming distance verification

We consider a set of k unperturbed inputs $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_k}\}$ where for all $i \in [k]$, $\mathbf{x_i} \in \mathbb{R}^{n_0}$, a peturbation budget $\epsilon \in \mathbb{R}$, and a binary digit classifier neural network $N_2 : \mathbb{R}^{n_0} \to \mathbb{R}^2$. We can define a binary digit string $\mathbf{S}^* \in \{0, 1\}^k$ as a sequence of binary digits where each input $\mathbf{x_i}$ to N_2 is an image of a binary digit. We are interested in bounding the worst-case hamming distance between $\mathbf{S}$, the binary digit string classified by N_2, and $\mathbf{S}^*$ the actual binary digit string corresponding to the list of perturbed images $\forall i \in [k].\mathbf{x_i^*} = \mathbf{x_i} + \boldsymbol{\delta}$ s.t. $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$. Given these definitions, we can use the Φ, Ψ and $\mu(\boldsymbol{\delta})$ defined in section A.1 defined for k-UAP verification. In this case, the worst case hamming distance $\mathbf{M}_0(\Phi, \Psi)$ is defined as $\mathbf{M}_0(\Phi, \Psi) = k - \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\| \leq \epsilon} \mu(\boldsymbol{\delta})$.

B. Theorectical guarantees for cross-execution bound refinement

We obtain the theoretical guarantees of cross-execution bound refinement over n executions. Note that we do not show the theoretical guarantees for a pair of executions separately as it is just a special case with $n = 2$.

B.1. Theorectical guarantees for n of executions

B.1.1. Theorems for LP formulation

First, we show the correctness of the LP formulation in Eq. 3 or for pair of execution in Eq. 1 (Theorem B.2). We also show that for fixed linear approximations $\{(\mathbf{L_1}, b_1), \ldots, (\mathbf{L_n}, b_n)\}$ of N, the LP formulation is exact i.e. it always proves the absence of common adversarial perturbation if it does not exist (Theorem B.3). In this case, $\Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}) = \bigvee_{i=1}^{n}(\mathbf{c_i}^T\mathbf{y_i} \geq 0)$ where the outputs of N are $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$. Let, t^* be the optimal solution of the LP in Eq. 3.

Lemma B.1. $t^* = \min\limits_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max\limits_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i$.

Proof. $t^* = \min\limits_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\| \leq \epsilon} t(\boldsymbol{\delta})$ where if $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ then $t(\delta)$ satisfies the following constraints $t(\delta) \geq \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i$ for all $i \in [n]$ then $t(\delta) \geq \max\limits_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i$. Let, $l^* = \min\limits_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\| \leq \epsilon} \max\limits_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i$.

$$t^* \geq \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i = l^* \tag{9}$$

Next, we show that $l^* \geq t^*$. $l^* = \max_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}^*) + b_i$ for some $\boldsymbol{\delta}^*$ where $\boldsymbol{\delta}^* \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}^*\|_\infty \leq \epsilon$, then l^* satisfies the constraints $l^* \geq \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}^*) + b_i$ for all $i \in [n]$. Since l^* is a valid feasible solution of the LP in Eq. 3 then $l^* \geq t^*$ as t^* is the optimal solution of the LP.

$l^* \geq t^*$ and from Eq. 9 $l^* \leq t^*$ implies $l^* = t^*$. □

Theorem B.2. *For all $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, if for all $i \in [n]$, $\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \leq \mathbf{c_i}^T\mathbf{y_i}$ then $(t^* \geq 0) \implies (\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}))$ holds.*

Proof. Since, for all $i \in [n]$, $\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \leq \mathbf{c_i}^T\mathbf{y_i}$, for all $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, then $\min\limits_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max\limits_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \leq \min\limits_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max\limits_{1 \leq i \leq n} \mathbf{c_i}^T\mathbf{y_i}$

$$t^* = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \leq \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max_{1 \leq i \leq n} \mathbf{c_i}^T\mathbf{y_i} \quad \text{Using lemma B.1}$$

$$(t^* \geq 0) \implies \left(\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max_{1 \leq i \leq n} \mathbf{c_i}^T\mathbf{y_i}\right) \geq 0$$

$$(t^* \geq 0) \implies (\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}))$$

□

Theorem B.3. $\left(\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \bigvee_{i=1}^{n}(\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \geq 0)\right)$ *holds if and only if $t^* \geq 0$.*

Proof. From lemma B.1, $t^* = \min\limits_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max\limits_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i$.

$$\begin{aligned}(t^* \geq 0) &\implies \left(\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max_{1 \leq i \leq n} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i\right) \geq 0 \\ &\implies \left(\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \bigvee_{i=1}^{n}(\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \geq 0)\right) \end{aligned} \tag{10}$$

$$\begin{aligned}(t^* < 0) &\implies \left(\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \max_{1 \leq i \leq 2} \mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i\right) < 0 \\ &\implies \left(\exists \boldsymbol{\delta} \in \mathbb{R}^{n_0}. \bigwedge_{i=1}^{n}(\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i < 0) \bigwedge (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)\right) \\ \neg(t^* \geq 0) &\implies \neg\left(\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \bigvee_{i=1}^{n}(\mathbf{L_i}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \geq 0)\right) \end{aligned} \tag{11}$$

Using Eq. 10 and Eq. 11, $(t^* \geq 0) \iff \Big(\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \bigvee_{i=1}^{n} ({\mathbf{L_i}}^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i \geq 0)\Big)$. □

B.1.2. Details for computing the Lagrangian Dual

Next, we provide the details for computing the Lagrangian Dual of the LP formulation in Eq. 3. The Lagrangian Dual is as follows where for all $i \in [n]$, $\lambda_i \geq 0$ are Lagrange multipliers.

$$\max_{0 \leq \lambda_i} \min_{t \in \mathbb{R}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} (1 - \sum_{i=1}^{n} \lambda_i) \times t + \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i\right)$$

We set the coefficient of the unbounded variable t to 0 to avoid cases where $\min_{t \in \mathbb{R}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} (1 - \sum_{i=1}^{n} \lambda_i) \times t + \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i\right) = -\infty$. This leads to the following Lagrangian Dual form

$$\max_{0 \leq \lambda_i} \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i^T(\mathbf{x_i} + \boldsymbol{\delta}) + b_i\right) \quad \text{where} \sum_{i=1}^{n} \lambda_i = 1$$

For all $i \in [n]$, let parametric linear approximations of N are specified by $(\mathbf{L}_i(\boldsymbol{\alpha}_i), \mathbf{b}_i(\boldsymbol{\alpha}_i))$ then the Lagrangian Dual is as follows

$$\max_{0 \leq \lambda_i} \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right) \quad \text{where} \sum_{i=1}^{n} \lambda_i = 1$$

B.1.3. Theorems for cross-execution bound refinement over n of executions

Let, the $t^*_{appx}(G)$ denote the solution obtained by the optimization technique and $\boldsymbol{\lambda}^*_{appx}$ denote the value of $\boldsymbol{\lambda}$ corresponding to $t^*_{appx}(G)$. Note that $t^*_{appx}(G)$ can be different from global maximum $t^*(G)$ with $t^*(G) > t^*_{appx}(G)$. We show that if $t^*_{appx}(G) \geq 0$ then $\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1}, \ldots \mathbf{y_n})$ holds where $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$ for all $i \in [n]$. First, we prove the correctness of the characterization of $G(\boldsymbol{\lambda})$.

Lemma B.4. *For all* $i \in [n]$, $0 \leq \lambda_i \leq 1$, $\sum_{i=1}^{n} \lambda_i = 1$, $\mathbf{l}_i \preceq \boldsymbol{\alpha}_i \preceq \mathbf{u}_i$, *if* $\boldsymbol{\lambda} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n, \lambda_1, \ldots, \lambda_n)$ *then* $\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies (G(\boldsymbol{\lambda}) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right)$ *where* $G(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) - \epsilon \times \sum_{j=1}^{n_0} \left| \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right|$ *and* $a_i(\boldsymbol{\alpha}_i) = \mathbf{L}_i(\boldsymbol{\alpha}_i)^T \mathbf{x_i} + \mathbf{b}_i(\boldsymbol{\alpha}_i)$.

Proof. First we rewrite $G(\boldsymbol{\lambda})$ in Eq. 12 and find the closed form on $\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T \boldsymbol{\delta}$ in Eq. 15.

$$\sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) + \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T \boldsymbol{\delta}$$

$$\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) + \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T \boldsymbol{\delta} \tag{12}$$

Now for fixed $\boldsymbol{\alpha}_i$, both $\mathbf{L}_i(\boldsymbol{\alpha}_i), \boldsymbol{\delta} \in \mathbb{R}^{n_0}$ are constant real vectors. Suppose for $j \in [n_0]$, $\mathbf{L}_i(\boldsymbol{\alpha}_i)[j]$ and $\boldsymbol{\delta}[j]$ denotes the j-th

component of $\mathbf{L}_i(\boldsymbol{\alpha}_i)$ and $\boldsymbol{\delta}$ respectively. Then,

$$\mathbf{L}_i(\boldsymbol{\alpha}_i)^T\boldsymbol{\delta} = \sum_{j=1}^{n_0} \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \times \boldsymbol{\delta}[j]$$

$$\sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T\boldsymbol{\delta} = \sum_{j=1}^{n_0} \left(\sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right) \times \boldsymbol{\delta}[j]$$

$$-\epsilon \times \left| \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right| = \min_{-\epsilon \le \boldsymbol{\delta}[j] \le \epsilon} \left(\sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right) \times \boldsymbol{\delta}[j] \tag{13}$$

$$\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \le \epsilon} \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T\boldsymbol{\delta} = \sum_{j=1}^{n_0} \min_{-\epsilon \le \boldsymbol{\delta}[j] \le \epsilon} \left(\sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right) \times \boldsymbol{\delta}[j] \tag{14}$$

$$\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \le \epsilon} \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T\boldsymbol{\delta} = -\epsilon \times \sum_{j=1}^{n_0} \left| \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right| \quad \text{using Eq 13 and Eq. 14} \tag{15}$$

Combing Eq. 12 and Eq. 15

$$\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \le \epsilon} \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i) \right) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) - \epsilon \times \sum_{j=1}^{n_0} \left| \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)[j] \right| = G(\boldsymbol{\lambda})$$

□

Theorem B.5 (Correctness of bound refinement over n executions)**.** *If* $t^*_{appx}(G) \ge 0$ *then* $(\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\| \le \epsilon) \implies \Psi(\mathbf{y_1}, \ldots \mathbf{y_n}))$ *holds where* $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$ *for all* $i \in [n]$.

Proof. $t^*_{appx}(G) = G(\boldsymbol{\lambda}^*_{appx})$ where $\boldsymbol{\lambda}^*_{appx} = (\boldsymbol{\alpha}^*_1, \ldots, \boldsymbol{\alpha}^*_n, \lambda^*_1, \ldots, \lambda^*_n)$ and for all $i \in [n]$, $\mathbf{l_i} \preceq \boldsymbol{\alpha}^*_i \preceq \mathbf{u_i}$, $0 \le \lambda^*_i \le 1$, $\sum_{i=1}^{n} \lambda^*_i = 1$. Then using lemma B.4 we get

$$G(\boldsymbol{\lambda}^*_{appx}) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \le \epsilon} \sum_{i=1}^{n} \lambda^*_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha^*}_i) \right) \tag{16}$$

Next we show that $G(\boldsymbol{\lambda}^*_{appx}) \le \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \le \epsilon} \max_{1 \le i \le n} \mathbf{c_i}^T\mathbf{y_i}$ where $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$.

$$\left(\mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha^*}_i) \right) \le \mathbf{c_i}^T\mathbf{y_i} \quad \forall i \in [n], \mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta}) \text{ and } \|\boldsymbol{\delta}\|_\infty \le \epsilon$$

$$\left(\mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha^*}_i) \right) \le \max_{1 \le i \le n} \mathbf{c_i}^T\mathbf{y_i} \qquad \forall i \in [n] \text{ and } \|\boldsymbol{\delta}\|_\infty \le \epsilon$$

$$\sum_{i=1}^{n} \lambda^*_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha^*}_i) \right) \le \max_{1 \le i \le n} \mathbf{c_i}^T\mathbf{y_i} \times \sum_{i=1}^{n} \lambda^*_i \qquad \text{since } \forall i \in [n], \lambda^*_i \ge 0 \text{ and } \|\boldsymbol{\delta}\|_\infty \le \epsilon$$

$$\sum_{i=1}^{n} \lambda^*_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha^*}_i) \right) \le \max_{1 \le i \le n} \mathbf{c_i}^T\mathbf{y_i} \qquad \text{since } \sum_{i=1}^{n} \lambda^*_i = 1 \text{ and } \|\boldsymbol{\delta}\|_\infty \le \epsilon$$

$$G(\boldsymbol{\lambda}^*_{appx}) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_\infty \le \epsilon} \sum_{i=1}^{n} \lambda^*_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha^*}_i) \right) \le \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \le \epsilon} \max_{1 \le i \le n} \mathbf{c_i}^T\mathbf{y_i} \tag{17}$$

Using Eq. 17 we show that

$$(t^*_{appx}(G) \ge 0) \implies (G(\boldsymbol{\lambda}^*_{appx}) \ge 0) \implies \left(\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \le \epsilon} \max_{1 \le i \le n} \mathbf{c_i}^T\mathbf{y_i} \right) \ge 0$$
$$\implies (\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \le \epsilon) \implies \Psi(\mathbf{y_1}, \ldots \mathbf{y_n}))$$

□

Similar to Theorem 4.2, we show the optimal solution $t^*(G)$ obtained with $G(\boldsymbol{\lambda})$ is always as good as $t^*(\overline{G})$ i.e. $t^*(G) \geq t^*(\overline{G})$ for n executions.

Theorem B.6. *If* $t^*(G) = \max_{\boldsymbol{\lambda}} G(\boldsymbol{\lambda})$ *and* $t^*(\overline{G}) = \max_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n} \overline{G}(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n)$ *then* $t^*(\overline{G}) \leq t^*(G)$.

Proof. For any $(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n)$ satisfying $\mathbf{l}_i \preceq \boldsymbol{\alpha}_i \preceq \mathbf{u}_i$ for all $i \in [n]$, we consider $\boldsymbol{\lambda}_i = (\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n, \lambda_1 = 0,\ldots,\lambda_i = 1,\ldots,\lambda_n = 0)$. Then $G(\boldsymbol{\lambda}_i) = \min_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} \mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)$. Since, $t^*(G) \geq G(\boldsymbol{\lambda}_i)$ for all $i \in [n]$ then $t^*(G) \geq \max_{1\leq i\leq n} G(\boldsymbol{\lambda}_i) = \overline{G}(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n)$. Hence, $t^*(G) \geq \max_{\boldsymbol{\alpha}_1,\ldots\boldsymbol{\alpha}_n} \overline{G}(\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2) = t^*(\overline{G})$. □

Next, we characterize one sufficient condition where $t^*(G)$ is strictly better i.e. $t^*(G) > t^*(\overline{G})$. Note that Theorem B.7 shows one possible case where $t^*(G)$ is strictly better and not the only possible condition where $t^*(G) > t^*(\overline{G})$ i.e. it is not necessary hold if $t^*(G) > t^*(\overline{G})$. Let, $(\boldsymbol{\alpha}^*_1,\ldots,\boldsymbol{\alpha}^*_n)$ be the optimal parameters corresponding to $t^*(\overline{G})$.

Theorem B.7. *If for all* $i \in [n]$ *there exists* $j \in [n]$ *such that* $(a_j(\boldsymbol{\alpha^*}_j) - a_i(\boldsymbol{\alpha^*}_i)) > \epsilon \times (\|\mathbf{L}_j(\boldsymbol{\alpha^*}_j)\|_1 - \|\mathbf{L}_i(\boldsymbol{\alpha^*}_i)\|_1)$ *or* $2 \times \|\mathbf{L}_i(\boldsymbol{\alpha^*}_i)\|_1 - \|\mathbf{L}_i(\boldsymbol{\alpha^*}_i) + \mathbf{L}_j(\boldsymbol{\alpha^*}_j)\|_1 > \frac{a_i(\boldsymbol{\alpha^*}_i)}{\epsilon} - \frac{a_j(\boldsymbol{\alpha^*}_j)}{\epsilon}$ *holds then* $t^*(G) > t^*(\overline{G})$.

Proof. Since $t^*(\overline{G}) = \max_{1\leq i\leq k} \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \mathbf{L}_i(\boldsymbol{\alpha^*}_i)^T\boldsymbol{\delta} + a_i(\boldsymbol{\alpha^*}_i) = \max_{1\leq i\leq k} -\epsilon \times \left(\sum_{j=1}^{n_0} |\mathbf{L}_i(\boldsymbol{\alpha^*}_i)[j]|\right) + a_i(\boldsymbol{\alpha^*}_i)$. This implies $t^*(\overline{G}) = \max_{1\leq i\leq k} -\epsilon \times \|\mathbf{L}_i(\boldsymbol{\alpha^*}_i)\|_1 + a_i(\boldsymbol{\alpha^*}_i)$. Now for any $i_0 \in [n]$ if $t^*(\overline{G}) = -\epsilon \times \|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0})\|_1 + a_{i_0}(\boldsymbol{\alpha^*}_{i_0})$ (there exists at least one such i_0) then

$$
\begin{aligned}
&-\epsilon \times \|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0})\|_1 + a_{i_0}(\boldsymbol{\alpha^*}_{i_0}) \geq -\epsilon \times \|\mathbf{L}_j(\boldsymbol{\alpha^*}_j)\|_1 + a_j(\boldsymbol{\alpha^*}_j) \quad \forall j \in [n] \\
&2 \times \|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0})\|_1 - \|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0}) + \mathbf{L}_{j_0}(\boldsymbol{\alpha^*}_{j_0})\|_1 > \frac{a_{i_0}(\boldsymbol{\alpha^*}_{i_0})}{\epsilon} - \frac{a_{j_0}(\boldsymbol{\alpha^*}_{j_0})}{\epsilon} \quad \text{for some } j_0 \in [n] \\
&\frac{1}{2} \times (-\epsilon \times (\|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0}) + \mathbf{L}_{j_0}(\boldsymbol{\alpha^*}_{j_0})\|_1) + a_{i_0}(\boldsymbol{\alpha^*}_{i_0}) + a_{j_0}(\boldsymbol{\alpha^*}_{j_0})) > -\epsilon \times \|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0})\|_1 + a_{i_0}(\boldsymbol{\alpha^*}_{i_0}) = t^*(\overline{G}) \quad (18)
\end{aligned}
$$

$t^*(G) = \max_{\boldsymbol{\lambda}} G(\boldsymbol{\lambda})$ now consider $\overline{\boldsymbol{\lambda}} = (\boldsymbol{\alpha}^*_1,\ldots,\boldsymbol{\alpha}^*_m, \lambda_1 = 0,\ldots,\lambda_{i_0} = \frac{1}{2},\ldots,\lambda_{j_0} = \frac{1}{2},\ldots\lambda_n = 0)$

$$
\begin{aligned}
t^*(G) \geq G(\overline{\boldsymbol{\lambda}}) &= \frac{1}{2} \times (-\epsilon \times (\|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0}) + \mathbf{L}_{j_0}(\boldsymbol{\alpha^*}_{j_0})\|_1) + a_{i_0}(\boldsymbol{\alpha^*}_{i_0}) + a_{j_0}(\boldsymbol{\alpha^*}_{j_0})) \\
t^*(G) &> -\epsilon \times \|\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0})\|_1 + a_{i_0}(\boldsymbol{\alpha^*}_{i_0}) = t^*(\overline{G}) \quad \text{Using Eq. 18}
\end{aligned}
$$

□

One simple example where this sufficient condition holds is $a_i(\boldsymbol{\alpha^*}_i) = a_j(\boldsymbol{\alpha^*}_j) = 0$ and $\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0}) = -\mathbf{L}_{j_0}(\boldsymbol{\alpha^*}_{j_0})$ and $-\mathbf{L}_{i_0}(\boldsymbol{\alpha^*}_{i_0})$ and $-\mathbf{L}_{j_0}(\boldsymbol{\alpha^*}_{j_0})$ are non-zero vectors.

B.2. Cross-execution bound refinement for conjunction of linear inequalities

We consider n executions of N on perturbed inputs given by $\{\mathbf{x_1} + \boldsymbol{\delta},\ldots,\mathbf{x_n} + \boldsymbol{\delta}\}$. In this case, to prove the absence of **common** adversarial perturbation we need to show for all $i \in [n]$ the outputs $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$ satisfy $\Psi(\mathbf{y_1},\ldots,\mathbf{y_n}) = \bigvee_{i=1}^n \psi^i(\mathbf{y_i})$. Here, $\psi^i(\mathbf{y_i}) = \bigwedge_{j=1}^m (\mathbf{c_{i,j}}^T\mathbf{y_i} \geq 0)$ and $\mathbf{c_{i,j}} \in \mathbb{R}^{n_l}$. First, we prove lemmas necessary for characterizing the optimizable closed form that can be used for bound refinement.

Lemma B.8. $\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.((\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1},\ldots,\mathbf{y_n}))$ *if and only if* $\left(\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1\leq i\leq n} \min_{1\leq j\leq m} \mathbf{c_{i,j}}^T\mathbf{y_i}\right) \geq 0$ *where for all* $i \in [n]$, $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$, $\Psi(\mathbf{y_1},\ldots,\mathbf{y_n}) = \bigvee_{i=1}^n \psi^i(\mathbf{y_i})$ *and* $\psi^i(\mathbf{y_i}) = \bigwedge_{j=1}^m (\mathbf{c_{i,j}}^T\mathbf{y_i} \geq 0)$.

Proof. We first show if $\left(\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\right)\geq 0$ then $\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.((\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\Psi(\mathbf{y_1},\ldots,\mathbf{y_n}))$.

$$\begin{aligned}\left(\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\right)\geq 0 &\implies (\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies(\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i})\geq 0)\\ &\implies(\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\vee_{i=1}^n((\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i})\geq 0))\\ &\implies\left(\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\vee_{i=1}^n\wedge_{j=1}^m(\mathbf{c_{i,j}}^T\mathbf{y_i}\geq 0)\right)\\ &\implies(\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.((\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\Psi(\mathbf{y_1},\ldots,\mathbf{y_n})))\end{aligned}\tag{19}$$

Next, we show if $\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.((\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\Psi(\mathbf{y_1},\ldots,\mathbf{y_n}))$ then $\left(\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\right)\geq 0$.

$$\begin{aligned}\left(\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\right)<0 &\implies(\exists\boldsymbol{\delta}\in\mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\wedge((\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i})<0))\\ &\implies(\exists\boldsymbol{\delta}\in\mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\wedge\neg(\vee_{i=1}^n\psi^i(\mathbf{y_i})))\\ &\implies\neg(\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\Psi(\mathbf{y_1},\ldots,\mathbf{y_n}))\end{aligned}\tag{20}$$

Eq. 20 is equivalent to showing the following

$$(\forall\boldsymbol{\delta}\in\mathbb{R}^{n_0}.((\|\boldsymbol{\delta}\|_\infty\leq\epsilon)\implies\Psi(\mathbf{y_1},\ldots,\mathbf{y_n})))\implies\left(\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\right)\geq 0$$

□

Lemma B.9. $\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}=\min_{j_1\in[m],\ldots,j_n\in[m]}S(j_1,\ldots,j_n)$ *where for all* $i\in[n]$ *and* $j_i\in[m]$ $S(j_1,\ldots,j_n)$ *is defined as* $S(j_1,\ldots,j_n)=\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\mathbf{c_{i,j_i}}^T\mathbf{y_i}$.

Proof. First, we show $\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\leq\min_{j_1\in[m],\ldots,j_n\in[m]}S(j_1,\ldots,j_n)$.

$$\begin{aligned}&\mathbf{c_{i,j_i}}^T\mathbf{y_i}\geq\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\quad\forall i\in[n]\text{ and }\forall j_i\in[m]\\ &S(j_1,\ldots,j_n)=\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\mathbf{c_{i,j_i}}^T\mathbf{y_i}\geq\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\quad\forall j_1\in[m],\ldots,j_n\in[m]\\ &\min_{j_1\in[m],\ldots,j_n\in[m]}S(j_1,\ldots,j_n)\geq\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\end{aligned}\tag{21}$$

Next, we show $\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\geq\min_{j_1\in[m],\ldots,j_n\in[m]}S(j_1,\ldots,j_n)$. There exists $\boldsymbol{\delta}^*\in\mathbb{R}^{n_0}$ such that $\|\boldsymbol{\delta}^*\|_\infty\leq\epsilon$, $\mathbf{y_i}^*=N(\mathbf{x_i}+\boldsymbol{\delta}^*)$ and $\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}^*=\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}$. Let, $j_i^*=\arg\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}^*$ then

$$\begin{aligned}&S(j_1^*,\ldots,j_n^*)=\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\mathbf{c_{i,j_i^*}}^T\mathbf{y_i}\\ &S(j_1^*,\ldots,j_n^*)\leq\max_{1\leq i\leq n}\mathbf{c_{i,j_i^*}}^T\mathbf{y_i}^*=\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}^*\quad\text{since }j_i^*=\arg\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}^*\\ &\min_{j_1\in[m],\ldots,j_n\in[m]}S(j_1,\ldots,j_n)\leq S(j_1^*,\ldots,j_n^*)\leq\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}\end{aligned}\tag{22}$$

Combining Eq. 21 and Eq. 22 we show $\min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},(\|\boldsymbol{\delta}\|_\infty\leq\epsilon)}\max_{1\leq i\leq n}\min_{1\leq j\leq m}\mathbf{c_{i,j}}^T\mathbf{y_i}=\min_{j_1\in[m],\ldots,j_n\in[m]}S(j_1,\ldots,j_n)$. □

Theorem B.10. $\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}. ((\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}))$ *if and only if* $\left(\min_{j_1 \in [m], \ldots, j_n \in [m]} S(j_1, \ldots, j_n)\right) \geq 0$ *where for all* $i \in [n]$, $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$, $\Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}) = \bigvee_{i=1}^{n} \psi^i(\mathbf{y_i})$, $\psi^i(\mathbf{y_i}) = \bigwedge_{j=1}^{m}(\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$ *and* $S(j_1, \ldots, j_n) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \mathbf{c_{i,j_i}}^T \mathbf{y_i}$.

Proof. Follows from lemma B.8 and lemma B.9. □

B.2.1. Reduction to bound refinement with single linear inequality

Theorem B.10 allows us to learn parameters for each $S(j_1, \ldots, j_n)$ separately so that $S(j_1, \ldots, j_n) \geq 0$ for each $(j_1, \ldots, j_n)$ where each $j_i \in [m]$. For $S(j_1, \ldots, j_n)$, let $\{(\mathbf{L}_{j_1}(\boldsymbol{\alpha}_{j_1}), \mathbf{b}_{j_1}(\boldsymbol{\alpha}_{j_1})), \ldots, (\mathbf{L}_{j_n}(\boldsymbol{\alpha}_{j_n}), \mathbf{b}_{j_n}(\boldsymbol{\alpha}_{j_n}))\}$ denote the linear approximations satisfying $\mathbf{L}_{j_i}(\boldsymbol{\alpha}_{j_i})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{j_i}(\boldsymbol{\alpha}_{j_i}) \leq \mathbf{c_{i,j_i}}^T \mathbf{y_i}$ for any $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ such that $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ and $\mathbf{l_{j_i}} \preceq \boldsymbol{\alpha}_{j_i} \preceq \mathbf{u_{j_i}}$. Then we can use cross-execution bound refinement for n executions to learn the parameters $(\boldsymbol{\alpha}_{j_1}, \ldots, \boldsymbol{\alpha}_{j_n})$. We repeat this process for all $(j_1, \ldots, j_n)$. However, the number of possible choices for $(j_1, \ldots, j_n)$ is m^n and learning parameters $(\boldsymbol{\alpha}_{j_1}, \ldots, \boldsymbol{\alpha}_{j_n})$ for all possible $(j_1, \ldots, j_n)$ is only practically feasible when both (m, n) are small constants. For larger values of (m, n) we greedily pick $(j_1, \ldots, j_n)$ for learning parameters to avoid the exponential blowup as detailed below.

Avoiding exponential blowup: Instead of learning parameters for all possible $(j_1, \ldots, j_n)$ we greedily select only single tuple $(j_1^*, \ldots, j_n^*)$. For the i-th execution with $\psi^i(\mathbf{y_i}) = \wedge_{i=1}^{m}(\mathbf{c_{i,j}}^T \mathbf{y_i} \geq 0)$, let $\{(\mathbf{L}_{i,1}(\boldsymbol{\alpha}^0_{i,1}), \mathbf{b}_{i,1}(\boldsymbol{\alpha}^0_{i,1})), \ldots, (\mathbf{L}_{i,m}(\boldsymbol{\alpha}^0_{i,m}), \mathbf{b}_{i,m}(\boldsymbol{\alpha}^0_{i,m}))\}$ dentoes linear approximations satisfying $\mathbf{L}_{i,j}(\boldsymbol{\alpha}^0_{i,j})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha}^0_{i,j}) \leq \mathbf{c_{i,j}}^T \mathbf{y_i}$ for all $j \in [m]$ and for all $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\| \leq \epsilon$. Note that for all $j \in [m]$, $\mathbf{l_i} \preceq \boldsymbol{\alpha}^0_{i,j} \preceq \mathbf{u_i}$ are the initial values of the parameters $\boldsymbol{\alpha}_{i,j}$. Now, for we select j_i^* for each execution as $j_i^* = \arg\min_{j \in [m]} \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \mathbf{L}_{i,j}(\boldsymbol{\alpha}^0_{i,j})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha}^0_{i,j})$.

Intuitively, we use j_i^* to determine the linear inequality $\mathbf{c_{i,j_i^*}}^T y_i \geq 0$ that is likely to be violated. For the tuple $(j_1^*, \ldots, j_n^*)$, let $\boldsymbol{\lambda}^*_{appx} = (\boldsymbol{\alpha}^*_{j_1^*}, \ldots, \boldsymbol{\alpha}^*_{j_n^*}, \lambda^*_{j_1^*}, \ldots, \lambda^*_{j_n^*})$ denote the learned parameters (which may not correspond to global optimum). Then we use the same parameters across all m linear approximations for the i-th execution i.e. $\{(\mathbf{L}_{i,1}(\boldsymbol{\alpha^*}_{j_i^*}), \mathbf{b}_{i,1}(\boldsymbol{\alpha^*}_{j_i^*})), \ldots, (\mathbf{L}_{i,m}(\boldsymbol{\alpha^*}_{j_i^*}), \mathbf{b}_{i,m}(\boldsymbol{\alpha^*}_{j_i^*}))\}$. In this case, $t^*_{appx}(G)$ is defined as $t^*_{appx}(G) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{L}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})$. Next, we prove the correctness of the bound refinement.

Theorem B.11 (Correctness of bound refinement for a conjunction of linear inequalities)**.** *If* $t^*_{appx}(G) \geq 0$ *then* $\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}. ((\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}))$ *where* $t^*_{appx}(G) = \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{L}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})$ *and for all* $i \in [n]$, $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$.

Proof. First we show that $t^*_{appx}(G) \leq \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i}$

$$
\begin{aligned}
&\mathbf{L}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*}) \leq \mathbf{c_{i,j}}^T \mathbf{y_i} \quad \forall i \in [n], \forall j \in [m] \text{ and for all } \boldsymbol{\delta} \in \mathbb{R}^{n_0} \text{ s.t } \|\boldsymbol{\delta}\|_\infty \leq \epsilon \\
&\min_{1 \leq j \leq m} \mathbf{L}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*}) \leq \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i} \quad \forall i \in [n] \text{ and for all } \boldsymbol{\delta} \in \mathbb{R}^{n_0} \text{ s.t } \|\boldsymbol{\delta}\|_\infty \leq \epsilon \\
&\max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{L}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*}) \leq \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i} \quad \text{for all } \boldsymbol{\delta} \in \mathbb{R}^{n_0} \text{ s.t } \|\boldsymbol{\delta}\|_\infty \leq \epsilon \\
&\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{L}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*})^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_{i,j}(\boldsymbol{\alpha^*}_{j_i^*}) \leq \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i} \\
&t^*_{appx}(G) \leq \min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i} \qquad (23)
\end{aligned}
$$

Using lemma B.8 and Eq 23

$$
\begin{aligned}
(t^*_{appx}(G) \geq 0) &\implies \left(\min_{\boldsymbol{\delta} \in \mathbb{R}^{n_0}, (\|\boldsymbol{\delta}\|_\infty \leq \epsilon)} \max_{1 \leq i \leq n} \min_{1 \leq j \leq m} \mathbf{c_{i,j}}^T \mathbf{y_i}\right) \geq 0 \\
&\implies \forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}. ((\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies \Psi(\mathbf{y_1}, \ldots, \mathbf{y_n}))
\end{aligned}
$$

□

B.3. Handling general $\|\cdot\|_p$ norm

For general $\|\cdot\|_p$ norm we can generalize the dual formulation $G(\boldsymbol{\lambda})$ in the following way. Since, $\|\boldsymbol{\delta}\|_p \leq \epsilon$ and $a_i(\boldsymbol{\alpha}_i) = \mathbf{L}_i(\boldsymbol{\alpha}_i)^T\mathbf{x_i} + \mathbf{b}_i(\boldsymbol{\alpha}_i)$ then

$$G(\boldsymbol{\lambda}) = \min_{\|\boldsymbol{\delta}\|_p \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \left(\mathbf{L}_i(\boldsymbol{\alpha}_i)^T(\mathbf{x_i} + \boldsymbol{\delta}) + \mathbf{b}_i(\boldsymbol{\alpha}_i)\right)$$

$$G(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) + \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0},\ \|\boldsymbol{\delta}\|_p \leq \epsilon} \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i)^T\boldsymbol{\delta}$$

$$G(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i \times a_i(\boldsymbol{\alpha}_i) - \epsilon \times \left\| \sum_{i=1}^{n} \lambda_i \times \mathbf{L}_i(\boldsymbol{\alpha}_i) \right\|_q \quad \text{Using Hölder's Inequality with } \frac{1}{q} = 1 - \frac{1}{p}$$

B.4. Optimization step details

We provide detailed descriptions for optimizing both $G(\boldsymbol{\lambda})$ and $\overline{G}(\overline{\boldsymbol{\alpha}})$ (From Eq. 1).

Optimizing $G(\boldsymbol{\lambda})$: In this case, $\boldsymbol{\lambda} = (\boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_k}, \lambda_1, \ldots, \lambda_k)$, $\forall i \in [k]\ \boldsymbol{l_i} \preceq \boldsymbol{\alpha_i} \preceq \boldsymbol{u_i}$, $\lambda_i \in [0,1]$ and $\sum_{i=1}^{k} \lambda_i = 1$.

For each $\boldsymbol{\alpha_i}$, after each step of gradient ascent (for maximization problem), we clip $\boldsymbol{\alpha_i}$ values to the range $[\boldsymbol{l_i}, \boldsymbol{u_i}]$. This is similar to the approach used in the SOTA non-relational bound refinement α-CROWN (Xu et al., 2021). Since $\lambda_i \in [0,1]$ and $\sum_{i=1}^{k} \lambda_i = 1$ we replace $\lambda_i = \frac{sigmoid(x_i)}{\sum_{i=1}^{k} sigmoid(x_i)}$ where $x_i \in \mathbb{R}$. For any values of $(x_1, \ldots, x_k) \in \mathbb{R}^k$ the corresponding $(\lambda_1, \ldots, \lambda_k)$ satisfy $\lambda_i \in [0,1]$ and $\sum_{i=1}^{k} \lambda_i = 1$. We then apply gradient ascent (for maximization problem) on $(x_1, \ldots, x_k)$. Note that the correctness of bound refinement is maintained for any $\boldsymbol{\lambda}$ satisfying $\forall i \in [k]\ \boldsymbol{l_i} \preceq \boldsymbol{\alpha_i} \preceq \boldsymbol{u_i}$, $\lambda_i \in [0,1]$ and $\sum_{i=1}^{k} \lambda_i = 1$. The aforementioned solution is one of the possible ways to solve the optimization problem with gradient ascent and is specific to our implementation. Any alternative solution is also acceptable provided the output $\boldsymbol{\lambda}$ satisfies the conditions.

Optimizing $\overline{G}(\overline{\boldsymbol{\alpha}})$: Since the naive formulation that uses $\overline{G}$ only includes $\boldsymbol{\alpha_i}$ s, the optimization problem can be solved using only step 1 described above.

B.5. MILP formulations and correctness

In this section, we show the MILP formulations for the k-UAP and worst-case hamming distance verification and present the theoretical results corresponding to the correctness and efficacy of the MILP formulations.

Let $I = \{i \mid \text{non-relational verifier does not verify } (\phi^i, \psi^i)\}$ denotes the executions that remain unverified by the non-relational verifier. For all $i \in I$, $j \in [m]$ let $(\mathbf{L}_{i,j}^{k'}, b_{i,j}^{k'})$ denote the linear approximations satisfying $\mathbf{L}_{i,j}^{k'}(\mathbf{x_i} + \boldsymbol{\delta}) + b_{i,j}^{k'} \leq \mathbf{c_{i,j}}^T\mathbf{y_i}$ for all $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ where $k' \leq \sum_{i=1}^{k_1} \binom{k_0}{i} + 1$ and $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$. Note that each linear approximations $(\mathbf{L}_{i,j}^{k'}, b_{i,j}^{k'})$ are obtained by the non-relational verifier or by the cross-execution bound refinement.

B.5.1. MILP FORMULATIONS

MILP formulation for k-UAP:

$$\begin{aligned}
&\min \quad M \\
&\|\boldsymbol{\delta}\|_\infty \leq \epsilon \\
&\mathbf{L}_{i,j}^{k'}(\mathbf{x_i} + \boldsymbol{\delta}) + b_{i,j}^{k'} \leq o_{i,j} \quad \forall i \in I, \forall j \in [m]\ \forall k' \\
&z_i = \left(\left(\min_{j\in[m]} o_{i,j}\right) \geq 0\right) \quad \text{for all } i \in I\ z_i \in \{0,1\} \\
&\overline{k} = k - |I| \quad \text{[number of executions verified by non-relational verifier]} \\
&M = \sum_{i\in I} z_i + \overline{k}
\end{aligned} \tag{24}$$

MILP formulation for worst-case hamming distance:

$$\begin{aligned}
&\max \quad M \\
&\\
&\|\boldsymbol{\delta}\|_\infty \leq \epsilon \\
&\mathbf{L}^{k'}_{i,j}(\mathbf{x_i} + \boldsymbol{\delta}) + b^{k'}_{i,j} \leq o_{i,j} \quad \forall i \in I, \forall j \in [m]\ \forall k' \\
&z_i = \left(\left(\min_{j\in[m]} o_{i,j}\right) \geq 0\right) \quad \text{for all } i \in I\ z_i \in \{0,1\} \\
&M = |I| - \sum_{i\in I} z_i
\end{aligned}$$

Correctness for eliminating individually verified executions: We formally prove that eliminating individually verified executions is correct and does not lead to precision loss.

Theorem B.12. $\mathbf{M}_0(\Phi,\Psi) = (k - |I|) + \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i\in I} z_i(\boldsymbol{\delta})$ *where* $z_i(\boldsymbol{\delta})$ *is defined in Eq. 6,* $\mathbf{M}_0(\Phi,\Psi)$ *is defined in Eq. 8 and for all* $j \in [k] \setminus I$, $\forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies (z_j(\delta) = 1)$ *holds.*

Proof.

$$\begin{aligned}
\mathbf{M}_0(\Phi,\Psi) &= \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i=1}^{k} z_i(\boldsymbol{\delta}) \\
&= \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i\in([k]\setminus I)} z_i(\boldsymbol{\delta}) + \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i\in I} z_i(\boldsymbol{\delta}) \\
&= (k - |I|) + \min_{\boldsymbol{\delta}\in\mathbb{R}^{n_0}, \|\boldsymbol{\delta}\|_\infty \leq \epsilon} \sum_{i\in I} z_i(\boldsymbol{\delta}) \quad \text{since } \forall \boldsymbol{\delta} \in \mathbb{R}^{n_0}.(\|\boldsymbol{\delta}\|_\infty \leq \epsilon) \implies (z_j(\delta) = 1)
\end{aligned}$$

□

Soundness of MILP formulation: For soundness, we show that the optimal value of the MILP formulation (in Eq. 24) $\mathbf{M}(\Phi,\Psi)$ is always a valid lower bound of $\mathbf{M}_0(\Phi,\Psi)$. The soundness of the worst-case hamming distance formulation can be proved similarly.

Theorem B.13 (Sondness of the MILP formulation in Eq. 24)**.** $\mathbf{M}(\Phi,\Psi) \leq \mathbf{M}_0(\Phi,\Psi)$ *where* $\mathbf{M}(\Phi,\Psi)$ *is the optimal solution of the MILP in Eq. 24 and* $\mathbf{M}_0(\Phi,\Psi)$ *is defined in Eq. 8.*

Proof. We prove this by contradiction. Suppose, $\mathbf{M}(\Phi,\Psi) > \mathbf{M}_0(\Phi,\Psi)$ then there exists $\boldsymbol{\delta}^* \in \mathbb{R}^{n_0}$ such that $\|\boldsymbol{\delta}^*\|_\infty \leq \epsilon$ and $\mathbf{M}(\Phi,\Psi) > \mu(\boldsymbol{\delta}^*)$ where $\mu(\boldsymbol{\delta})$ defined in Eq. 7.

For all $i \in I$, $j \in [m]$ the linear approximation $(\mathbf{L}^{k'}_{i,j}, b^{k'}_{i,j})$ satisfies $\mathbf{L}^{k'}_{i,j}(\mathbf{x_i} + \boldsymbol{\delta}) + b^{k'}_{i,j} \leq \mathbf{c_{i,j}}^T \mathbf{y_i}$ for all $\boldsymbol{\delta} \in \mathbb{R}^{n_0}$ and $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ where $k' \leq \sum_{i=1}^{k_1} \binom{k_0}{i} + 1$ and $\mathbf{y_i} = N(\mathbf{x_i} + \boldsymbol{\delta})$. Let, $z^*_i(\boldsymbol{\delta}^*) = \left(\min_{j\in[m]} o^*_{i,j}(\boldsymbol{\delta}^*) \geq 0\right)$ where $o^*_{i,j}(\boldsymbol{\delta}^*) = \max_{k'} \mathbf{L}^{k'}_{i,j}(\mathbf{x_i} + \boldsymbol{\delta}^*) + b^{k'}_{i,j}$. Then $\mathbf{M}(\Phi,\Psi) \leq \overline{k} + \sum_{i\in I} z^*_i(\boldsymbol{\delta}^*)$ and $\mu(\boldsymbol{\delta}^*) < \overline{k} + \sum_{i\in I} z^*_i(\boldsymbol{\delta}^*)$.

$$\begin{aligned}
&\mu(\boldsymbol{\delta}^*) < \overline{k} + \sum_{i\in I} z^*_i(\boldsymbol{\delta}^*) \\
&\implies \sum_{i\in I} z_i(\boldsymbol{\delta}^*) < \sum_{i\in I} z^*_i(\boldsymbol{\delta}^*) \quad \text{where } z_i(\boldsymbol{\delta}^*) \text{ defined in Eq. 6}
\end{aligned} \tag{25}$$

Eq. 25 implies that there exist $i_0 \in I$ such that $z_{i_0}(\boldsymbol{\delta}^*) = 0$ and $z^*_{i_0}(\boldsymbol{\delta}^*) = 1$. Since $z_{i_0}(\boldsymbol{\delta}^*) = 0$ then there exists $j_0 \in [m]$ such that $\mathbf{c_{i_0,j_0}}^T \mathbf{y_{i_0}}^* < 0$ where $\mathbf{y_{i_0}}^* = N(\mathbf{x_{i_0}} + \boldsymbol{\delta}^*)$

$$\begin{aligned}
&\min_{j\in[m]} o^*_{i_0,j}(\boldsymbol{\delta}^*) \leq o^*_{i_0,j_0}(\boldsymbol{\delta}^*) \leq \mathbf{c_{i_0,j_0}}^T \mathbf{y_{i_0}}^* < 0 \\
&(\min_{j\in[m]} o^*_{i_0,j}(\boldsymbol{\delta}^*) < 0) \implies (z^*_{i_0}(\boldsymbol{\delta}^*) = 0) \quad \text{Contradiction since } z^*_{i_0}(\boldsymbol{\delta}^*) = 1
\end{aligned}$$

□

Next, we show that RACoon is always at least as precise as the current SOTA relational verifier (Zeng et al., 2023). Note that (Zeng et al., 2023) uses the same MILP formulation (Eq. 24) except instead of using k' linear approximations $\{(\mathbf{L}^1_{i,j}, b^1_{i,j}), \ldots, (\mathbf{L}^{k'}_{i,j}, b^{k'}_{i,j})\}$ it uses a single statically obtained linear approximation say $\{(\mathbf{L}^1_{i,j}, b^1_{i,j})\}$.

Theorem B.14 (RACoon is at least as precise as (Zeng et al., 2023))**.** *$\mathbf{M}_b(\Phi, \Psi) \leq \mathbf{M}(\Phi, \Psi)$ where $\mathbf{M}(\Phi, \Psi)$ is the optimal solution of the MILP in Eq. 24 and $\mathbf{M}_b(\Phi, \Psi)$ is the optimal solution from the baseline (Zeng et al., 2023).*

Proof. Now we show that for $i \in I, \forall j \in [m]$ for every feasible value of the variable $o_{i,j}$ in Eq. 24 is also a feasible value of the same variable $o_{i,j}$ in MILP of (Zeng et al., 2023). Given $\forall k'$, $\mathbf{L}^{k'}_{i,j}(\mathbf{x_i} + \boldsymbol{\delta}) + b^{k'}_{i,j} \leq o_{i,j}$ then trivally $o_{i,j}$ satisfies condition $\mathbf{L}^1_{i,j}(\mathbf{x_i} + \boldsymbol{\delta}) + b^1_{i,j} \leq o_{i,j}$ used by the baseline (Zeng et al., 2023). Subsequently for all $i \in I$ every feasible value of z_i in Eq. 24 is also a feasible value of the same variable z_i in the MILP of (Zeng et al., 2023). Let. for all $i \in I$, $\mathcal{Z}$ and $\mathcal{Z}_b$ denote the sets of all feasible values of variables $(z_1, \ldots, z_I)$ from the MILP in Eq. 24 and the baseline (Zeng et al., 2023) respectively. Then $\mathcal{Z} \subseteq \mathcal{Z}_b$ which implies

$$\begin{aligned}\mathbf{M}_b(\Phi, \Psi) &\leq k - |I| + \min_{(z_1,\ldots,z_I)\in\mathcal{Z}_b} \sum_{i\in I} z_i \\ &\leq k - |I| + \min_{(z_1,\ldots,z_I)\in\mathcal{Z}} \sum_{i\in I} z_i = \mathbf{M}(\Phi, \Psi) \quad \text{Since } \mathcal{Z} \subseteq \mathcal{Z}_b\end{aligned}$$

□

C. Worst-case time complexity analysis of RACoon

Let, the total number of neurons in N be n_t and the number of layers in N is l. Then for each execution, the worst-case cost of running the non-relational verifier (Xu et al., 2020) is $O(l^2 \times n_t^3)$. We assume that we run I_t number of iterations with the optimizer and the cost of each optimization step over a set of n executions is $O(n \times C_o)$. In general, C_o is similar to the cost of the non-relational verifier i.e. $O(l^2 \times n_t^3)$. Then the total cost of cross-execution refinement is $O(T \times I_t \times C_o)$ where $T = \sum_{i=1}^{k_1} \left(\binom{k_0}{i} \times i\right)$. Assuming MILP with $O(k \times n_l)$ integer variables in the worst-case takes $C_M(k \times n_l)$ time. Then the worst-case complexity of RACoon is $O(k \times l^2 \times n_t^3) + O(T \times I_t \times C_o) + C_M(k \times n_l)$.

D. Details of DNN archietectures

Table 2. DNN architecture details

Dataset	Model	Type	Train	# Layers	# Params
MNIST	IBPSmall	Conv	IBP	4	60k
	ConvSmall	Conv	Standard	4	80k
	ConvSmall	Conv	PGD	4	80k
	ConvSmall	Conv	DiffAI	4	80k
	ConvSmall	Conv	COLT	4	80k
	IBPMedium	Conv	IBP	5	400k
	ConvBig	Conv	DiffAI	7	1.8M
CIFAR10	IBP-Small	Conv	IBP	4	60k
	ConvSmall	Conv	Standard	4	80k
	ConvSmall	Conv	PGD	4	80k
	ConvSmall	Conv	DiffAI	4	80k
	ConvSmall	Conv	COLT	4	80k
	IBPMedium	Conv	IBP	5	2.2M
	ConvBig	Conv	DiffAI	7	2.5M

D.1. Implementation Details

We implemented our method in Python with Pytorch V1.11 and used Gurobi V10.0.3 as an off-the-shelf MILP solver. The implementation of cross-execution bound refinement is built on top of the SOTA DNN verification tool auto_LiRPA (Xu

et al., 2021) and uses Adam (Kingma & Ba, 2014) for parameter learning. We run 20 iterations of Adam on each set of executions. For each relational property, we use $k_0 = 6$ and $k_1 = 4$ for deciding which set of executions to consider for cross-execution refinement as discussed in section 5. We use a single NVIDIA A100-PCI GPU with 40 GB RAM for bound refinement and an Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz with 64 GB RAM for MILP optimization. For each network, we calculate the worst-case UAP accuracy only on images that are correctly classified. We filter out any misclassified images while computing the worst-case UAP accuracy.

E. UAP accuracy over data distribution

Table 3. Statistical estimation worst case UAP accuracy over input distribution using k-UAP accuracy different with different k values.

Dataset	Property	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier UAP Acc. (%)			I/O Formulation UAP Acc. (%)			RACoon UAP Acc. (%)		
					$k=20$	$k=30$	$k=50$	$k=20$	$k=30$	$k=50$	$k=20$	$k=30$	$k=50$
MNIST	UAP	ConvSmall	Standard	0.08	11.00	15.33	19.20	20.50	28.0	33.40	26.50 (+6.00)	31.00 (+3.00)	34.40 (+1.00)
	UAP	ConvSmall	PGD	0.10	43.00	46.00	47.80	44.50	49.00	53.00	49.50 (+5.00)	54.00 (+5.00)	57.00 (+4.00)
	UAP	IBPSmall	IBP	0.13	47.00	51.00	55.20	47.50	51.67	57.80	61.50 (+14.00)	67.67 (+16.00)	69.80 (+12.00)
	UAP	ConvSmall	DiffAI	0.13	28.50	31.00	35.20	33.50	38.67	46.20	40.50 (+7.00)	45.00 (+6.33)	48.60 (+2.40)
	UAP	ConvSmall	COLT	0.15	41.50	46.33	48.80	41.50	46.67	49.80	58.00 (+16.50)	63.00 (+16.33)	65.60 (+15.80)
Dataset	Property	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier UAP Acc. (%)			I/O Formulation UAP Acc. (%)			RACoon UAP Acc. (%)		
					$k=15$	$k=20$	$k=25$	$k=15$	$k=20$	$k=25$	$k=15$	$k=20$	$k=25$
CIFAR10	UAP	ConvSmall	Standard	1.0/255	16.93	19.00	21.90	22.27	27.00	30.70	24.27 (+2.00)	28.00 (+1.00)	32.30 (+1.60)
	UAP	ConvSmall	PGD	2.0/255	19.60	27.50	30.30	23.60	33.00	35.50	24.27 (+0.67)	33.50 (+0.50)	37.50 (+2.00)
	UAP	IBPSmall	IBP	3.0/255	23.60	31.50	34.30	23.60	31.50	35.10	34.27 (+10.67)	42.00 (+10.50)	46.30 (+11.20)
	UAP	ConvSmall	DiffAI	3.0/255	36.27	39.00	43.90	38.93	45.50	50.70	40.93 (+2.00)	46.50 (+1.00)	51.50 (+0.80)
	UAP	ConvSmall	COLT	6.0/255	13.60	19.50	21.50	18.93	26.50	27.50	22.93 (+4.00)	29.00 (+2.50)	29.90 (+2.40)

All values in Table 3 are computed using Theorem 2 of (Zeng et al., 2023) with $\xi = 0.1$.

F. RACoon componentwise efficacy analysis on all DNNs

Table 4. RACoon Componentwise Efficacy Analysis

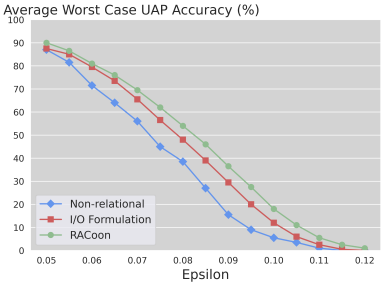
Dataset	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier Avg. UAP Acc. (%)	I/O Formulation Avg. UAP Acc. (%)	Individual Refinement Avg. UAP Acc. (%)	Individual Refinement with MILP Avg. UAP Acc. (%)	Cross-Execution Refinement Avg. UAP Acc. (%)	RACoon verifier Avg. UAP Acc. (%)
MNIST	ConvSmall	Standard	0.08	38.5	48.0	42.5	50.5	51.0	54.0
	ConvSmall	PGD	0.10	70.5	72.0	72.5	74.0	76.5	77.0
	IBPSmall	IBP	0.13	74.5	75.0	84.0	84.5	89.0	89.0
	ConvSmall	DiffAI	0.13	56.0	61.0	61.0	64.5	67.0	68.0
	ConvSmall	COLT	0.15	69.0	69.0	72.5	72.5	81.5	85.5
	IBPMedium	IBP	0.20	80.5	82.0	91.0	91.0	93.5	93.5
	ConvBig	DiffAI	0.20	80.0	80.0	86.0	86.0	90.0	93.0
CIFAR10	ConvSmall	Standard	1.0/255	52.0	55.0	52.0	55.0	57.0	58.0
	ConvSmall	PGD	3.0/255	21.0	26.0	22.0	27.0	29.0	29.0
	IBPSmall	IBP	6.0/255	17.0	17.0	29.0	29.0	36.0	39.0
	ConvSmall	DiffAI	8.0/255	16.0	20.0	26.0	28.0	29.0	30.0
	ConvSmall	COLT	8.0/255	18.0	21.0	22.0	22.0	24.0	26.0
	IBPMedium	IBP	3.0/255	46.0	50.0	63.0	63.0	69.0	71.0
	ConvBig	DiffAI	3.0/255	17.0	20.0	24.0	25.0	25.0	25.0

G. RACoon componentwise runtime analysis on all DNNs

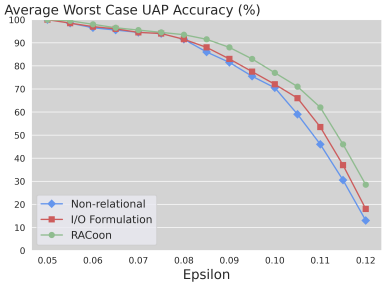
Table 5. RACoon Componentwise Runtime Analysis

Dataset	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier Avg. Time (sec.)	I/O Formulation Avg. Time (sec.)	Individual Refinement Avg. Time (sec.)	Individual Refinement with MILP Avg. Time (sec.)	Cross-Execution Refinement Avg. Time (sec.)	RACoon verifier Avg. Time (sec.)
MNIST	ConvSmall	Standard	0.08	0.02	2.66	0.58	3.21	3.42	5.21
	ConvSmall	PGD	0.10	0.02	0.93	0.61	1.82	3.47	4.33
	IBPSmall	IBP	0.13	0.02	1.02	0.48	1.78	1.58	2.02
	ConvSmall	DiffAI	0.13	0.01	1.10	0.52	2.11	2.84	3.99
	ConvSmall	COLT	0.15	0.02	0.99	0.50	1.82	2.17	2.68
	IBPMedium	IBP	0.20	0.07	0.99	0.90	2.02	1.91	2.27
	ConvBig	DiffAI	0.20	1.85	2.23	3.70	4.07	7.36	7.60
CIFAR10	ConvSmall	Standard	1.0/255	0.02	3.46	0.50	5.48	2.99	7.22
	ConvSmall	PGD	3.0/255	0.01	1.57	0.40	3.64	2.44	5.56
	IBPSmall	IBP	6.0/255	0.02	2.76	0.56	3.92	3.32	6.76
	ConvSmall	DiffAI	8.0/255	0.01	2.49	0.49	4.75	2.96	7.09
	ConvSmall	COLT	8.0/255	0.04	2.41	0.92	3.95	6.73	11.02
	IBPMedium	IBP	3.0/255	0.15	2.13	1.77	4.07	5.28	6.12
	ConvBig	DiffAI	3.0/255	1.33	3.42	3.27	5.89	10.45	11.92

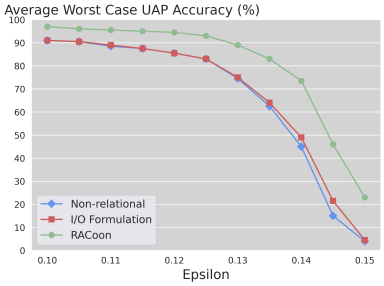
H. Additional plots for k-UAP for different ϵ values



. (a) ConvSmall (Standard)

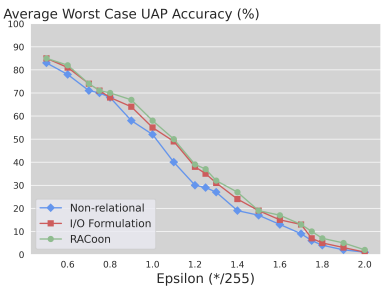


. (b) ConvSmall (PGD)

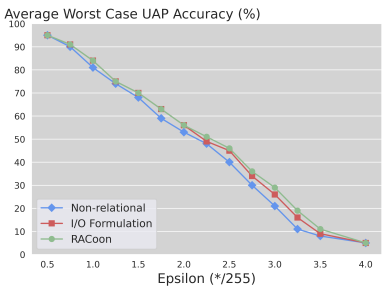


. (c) IBPSmall

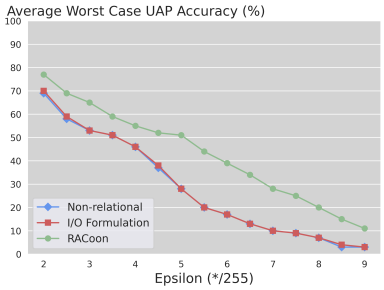
Figure 3. Average worst-case UAP accuracy for different ϵ values for networks trained on MNIST.



. (a) ConvSmall (Standard)



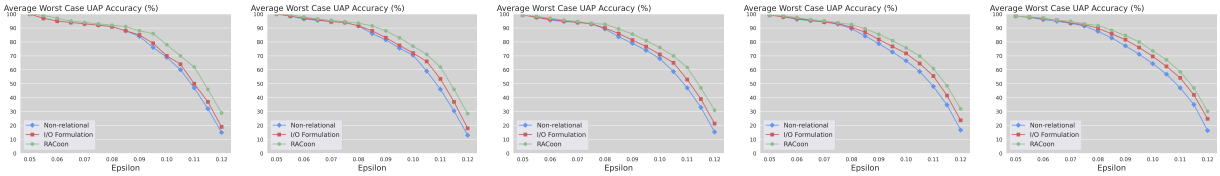
. (b) ConvSmall (PGD)



. (c) IBPSmall

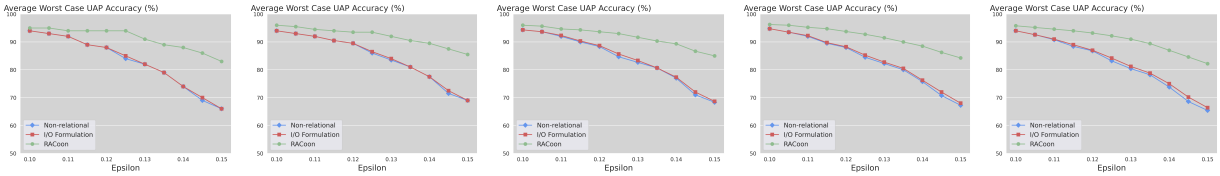
Figure 4. Average worst-case UAP accuracy for different ϵ values for networks trained on CIFAR10.

I. k-UAP verification results for different k and ϵ values



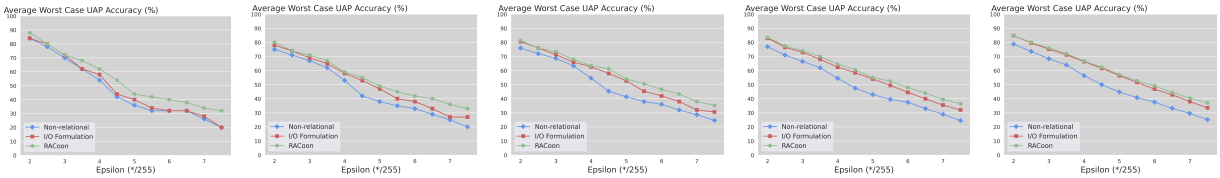
. (a) $k = 10$. (b) $k = 20$. (c) $k = 30$. (d) $k = 40$. (e) $k = 50$

Figure 5. Average worst-case UAP accuracy for different k and ϵ values for ConvSmall PGD MNIST network.



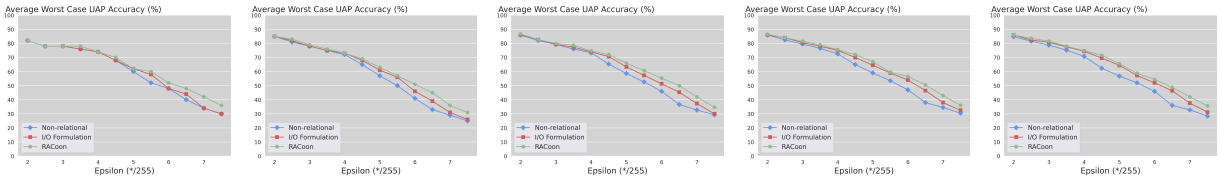
. (a) $k = 10$. (b) $k = 20$. (c) $k = 30$. (d) $k = 40$. (e) $k = 50$

Figure 6. Average worst-case UAP accuracy for different k and ϵ values for ConvSmall COLT MNIST network.



. (a) $k = 5$. (b) $k = 10$. (c) $k = 15$. (d) $k = 20$. (e) $k = 25$

Figure 7. Average worst-case UAP accuracy for different k and ϵ values for ConvSmall DiffAI CIFAR10 network.



. (a) $k = 5$. (b) $k = 10$. (c) $k = 15$. (d) $k = 20$. (e) $k = 25$

Figure 8. Average worst-case UAP accuracy for different k and ϵ values for ConvSmall COLT CIFAR10 network.

Table 6. RACoon Componentwise Runtime Analysis for $k = 50$ for MNIST and $k = 25$ for CIFAR10 networks

Dataset	Network Structure	Training Method	Perturbation Bound (ϵ)	Non-relational Verifier Avg. Time (sec.)	I/O Formulation Avg. Time (sec.)	Individual Refinement Avg. Time (sec.)	Individual Refinement with MILP Avg. Time (sec.)	Cross-Execution Refinement Avg. Time (sec.)	RACoon verifier Avg. Time (sec.)
MNIST	ConvSmall	Standard	0.08	0.01	31.76 *	0.40	24.11	2.25	4.16
	ConvSmall	PGD	0.10	0.02	5.13	0.50	5.36	3.00	4.59
	IBPSmall	IBP	0.13	0.01	3.62	0.39	3.38	1.53	2.27
	ConvSmall	DiffAI	0.13	0.02	18.32 *	0.59	9.89	3.34	5.38
	ConvSmall	COLT	0.15	0.04	2.53	0.43	3.41	1.07	1.64
	ConvBig	DiffAI	0.20	2.14	5.23	9.70	11.07	9.36	14.30
CIFAR10	ConvSmall	Standard	1.0/255	0.04	91.74 *	0.86	101.17	4.54	19.39
	ConvSmall	PGD	3.0/255	0.03	8.28	0.73	12.53	4.24	11.47
	IBPSmall	IBP	6.0/255	0.02	2.76	0.56	3.92	3.32	6.76
	ConvSmall	DiffAI	7.0/255	0.01	12.22 *	0.46	13.62	2.69	9.68
	ConvSmall	COLT	7.0/255	0.07	15.74	0.95	25.82	5.20	24.54
	ConvBig	DiffAI	3.0/255	2.01	13.42	7.27	22.89	16.45	21.92

* I/O formulation does not filter out executions verified by the non-relational verifier making MILP optimization for large k expensive.

J. Ablation study of the hyperparameters k_0 and k_1 on different MNIST networks

Table 7. RACoon Average worst-case UAP accuracy on ConvSmall Standard MNIST net with different k_0 and k_1 on $\epsilon = 0.1$.

k_1 \ k_0	2	3	4	5	6	7
2	26.50	28.50	31.50	32.00	33.00	34.50
3	×	28.50	31.50	32.00	33.00	34.50
4	×	×	32.00	33.00	33.00	34.50

Table 8. RACoon Average runtime on ConvSmall Standard MNIST net with different k_0 and k_1 on $\epsilon = 0.1$.

k_1 \ k_0	2	3	4	5	6	7
2	1.35	1.45	2.35	2.75	3.71	5.06
3	×	1.53	2.12	3.79	4.78	8.68
4	×	×	2.16	3.94	6.09	8.15

Table 9. RACoon Average worst-case UAP accuracy on ConvSmall PGD MNIST net with different k_0 and k_1 on $\epsilon = 0.1$.

k_1 \ k_0	2	3	4	5	6	7
2	54.50	56.50	58.00	59.50	60.50	61.00
3	×	56.50	58.50	60.00	61.50	62.00
4	×	×	58.50	60.00	61.50	62.00

Table 10. RACoon Average runtime on ConvSmall PGD MNIST net with different k_0 and k_1 on $\epsilon = 0.1$.

k_1 \ k_0	2	3	4	5	6	7
2	1.03	1.20	1.67	2.20	2.83	4.32
3	×	1.44	2.24	3.71	5.13	7.39
4	×	×	2.22	4.28	6.67	8.41

Table 11. RACoon Average worst-case UAP accuracy on ConvSmall DiffAI MNIST net with different k_0 and k_1 on $\epsilon = 0.12$.

k_1 \ k_0	2	3	4	5	6	7
2	83.50	84.50	85.00	85.00	85.00	85.00
3	×	84.50	85.00	85.00	85.00	85.00
4	×	×	85.00	85.00	85.00	85.00

Table 12. RACoon Average runtime on ConvSmall DiffAI MNIST net with different k_0 and k_1 on $\epsilon = 0.12$.

k_1 \ k_0	2	3	4	5	6	7
2	0.95	1.29	2.27	2.70	1.90	1.89
3	×	1.29	2.79	3.55	2.89	2.70
4	×	×	3.07	3.25	2.93	3.11

Table 13. RACoon Average worst-case UAP accuracy on IBPSmall MNIST net with different k_0 and k_1 on $\epsilon = 0.15$.

k_1 \ k_0	2	3	4	5	6	7
2	55.50	59.00	63.00	65.50	68.00	69.50
3	×	59.00	63.00	65.50	68.00	69.50
4	×	×	64.00	66.50	68.00	69.50

Table 14. RACoon Average runtime on IBPSmall MNIST net with different k_0 and k_1 on $\epsilon = 0.15$.

k_1 \ k_0	2	3	4	5	6	7
2	0.91	0.95	0.91	0.98	1.57	1.90
3	×	0.89	0.87	0.95	1.25	1.76
4	×	×	0.94	1.16	1.50	1.67