ModelStar: Reachability Analysis-based Safety Verification of Neural Networks Against Model Perturbations

MUHAMMAD USAMA ZUBAIR*, University of Texas at Dallas, USA
TAYLOR T. JOHNSON, Vanderbilt University, USA
KANAD BASU, University of Texas at Dallas, USA
WASEEM ABBAS, University of Texas at Dallas, USA

The widespread adoption of deep neural network (DNN)-based learning systems in safety-critical applications requires exceptional reliability. However, this reliability could be compromised by perturbations in model parameters, such as variations in neural network weights caused by hardware vulnerabilities and environmental factors, which can lead to mispredictions and compromise system safety. To address this, we propose 'ModelStar', an innovative framework leveraging reachability analysis to evaluate the robustness of DNNs against weight perturbations. ModelStar employs a linear set propagation technique to analyze the impact of an infinite family of parameter variations on DNN outputs. Our comprehensive analysis demonstrates that ModelStar not only establishes tighter robustness bounds but also verifies DNN robustness for up to 60% more samples from image classification datasets compared to existing methods. Furthermore, ModelStar extends safety verification to convolutional layers, advancing the state-of-the-art in neural network safety verification. These results highlight ModelStar's efficacy in improving the reliability of DNNs in real-world, safety-critical scenarios.

JAIR Track: Integration of Logical Constraints in Deep Learning

JAIR Associate Editor: Bettina Koenighofer

JAIR Reference Format:
Muhammad Usama Zubair, Taylor T. Johnson, Kanad Basu, and Waseem Abbas. 2026. ModelStar: Reachability Analysis-based Safety Verification of Neural Networks Against Model Perturbations. *Journal of Artificial Intelligence Research* 85, Article 37 (April 2026), 29 pages. DOI: 10.1613/jair.1.18922

1 Introduction

Neural network (NN)-based learning systems are increasingly employed in safety-critical applications, such as autonomous driving, collision avoidance systems, and medical diagnostics (Z. Li et al. 2021; Perez-Cerrolaza et al. 2024; Rech 2024; Yao et al. 2024). Reliability is paramount in these systems, as failures can result in catastrophic consequences, including fatal accidents involving autonomous vehicles (Levin and Wong 2018; Yadron and Tynan 2016). Recent studies on the vulnerabilities of NNs underscore the urgent need for rigorous evaluation of NN robustness, especially in environments where safety cannot be compromised (Korkmaz 2021; Michel et al. 2022; Smagulova et al. 2024; Sun et al. 2021; Wang et al. 2023; Yan et al. 2022). Although significant progress has been made in verifying robustness against input perturbations (Fang et al. 2024; Fazlyab et al. 2024; Grimm et al. 2024;

*Corresponding Author.

Authors' Contact Information: Muhammad Usama Zubair, ORCID: 0000-0002-0031-8671, muhammadusama.zubair@utdallas.edu, University of Texas at Dallas, Richardson, Texas, USA; Taylor T. Johnson, ORCID: 0000-0001-8021-9923, taylor.johnson@vanderbilt.edu, Vanderbilt University, Nashville, Tennessee, USA; Kanad Basu, ORCID: 0000-0002-6431-7512, kanad.basu@utdallas.edu, University of Texas at Dallas, Richardson, Texas, USA; Waseem Abbas, ORCID: 0000-0002-9013-1463, waseem.abbas@utdallas.edu, University of Texas at Dallas, Richardson, Texas, USA.

DOI: 10.1613/jair.1.18922

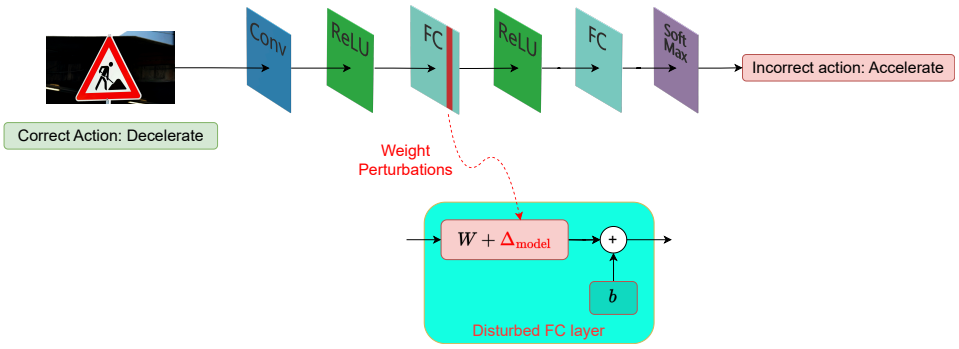

Fig. 1. Model perturbations in NNs may lead to fatal accidents in safety-critical systems. For example, an autonomous vehicle may accelerate instead of decelerating when it encounters a construction sign. Here, Δ_{model} represents the weight perturbation matrix, and the terms 'Conv' and 'FC' stand for convolutional layer and fully-connected layer, respectively.

L. Li et al. 2023; Liu et al. 2024; Meng et al. 2022; Wu et al. 2024; Zhao et al. 2024; Zühlke and Kudenko 2024), the vulnerability of NNs to weight perturbations—variations in trained model parameters—is comparatively less studied in the literature. Hardware imperfections in analog neural network accelerators (Shin et al. 2022; Xiao et al. 2022; Yan et al. 2022) and bit-level failures in digital neural network accelerators (Ganapathy et al. 2017; Maji et al. 2024; Rahman et al. 2024; Santos et al. 2025) highlight fundamental limitations that can induce weight perturbations. Furthermore, weight variations can also be induced by adversarial attacks (Cai et al. 2024; Fuengfusin and Tamukoh 2024; Gongye and Fei 2024; S. Li et al. 2024; Pourmehrani et al. 2024; Qian et al. 2023). These perturbations can significantly impact model predictions, posing a serious threat to reliability in real-world applications (Sun et al. 2021; Yan et al. 2022). For example, Figure 1 illustrates how an NN may mispredict a driving action due to weight perturbations. Consequently, developing reliable methods for verifying NN robustness against weight variations is essential for ensuring the safe deployment of NNs in safety-critical systems.

Recently, there has been an increased interest in providing guaranteed lower bounds on the robustness of NNs against weight perturbations. For instance, bounds on pairwise class margins are provided by Tsai et al. (2021a) to assess the impact of such perturbations. T.-W. Weng et al. (2020) extended linear bounds, originally designed for input perturbations (L. Weng et al. 2018), to address weight perturbations. Although these methods represent important advances, they often provide conservative robustness guarantees, leaving significant potential for tighter and more informative bounds.

In this work, we advance the state-of-the-art by *leveraging reachability analysis to improve robustness bounds against weight perturbations*. Traditionally, reachability analysis has been used to construct and propagate sets representing all possible states resulting from input perturbations through the layers of an NN. We extend this technique to handle weight perturbations by augmenting the state representations within these sets as they pass through layers with perturbed weights. This extension is crucial because it propagates richer information through the layers of an NN compared to previous approaches (Tsai et al. 2021a; T.-W. Weng et al. 2020), leading to significantly tighter robustness guarantees. To achieve this, we utilize the ImageStar reachability analysis framework (Tran, Bak, et al. 2020a), known for its robust support for propagating reachable sets across diverse NN architectures (Lopez et al. 2023). Our proposed extension integrates model perturbations directly into the star representation, resulting in a novel framework we term **ModelStar**. By providing tighter robustness guarantees, ModelStar enhances the reliability of NNs in practical safety-critical applications.

ModelStar introduces a novel approach to robustness analysis by representing variations in individual NN weights as continuous intervals. This representation enables the verification of safe NN operation across a

continuum of possible weight values, effectively ensuring the robustness of an infinite family of NNs with parameters confined to specified ranges. While this approach aligns with the concept of Interval Neural Networks (INNs) (Boudardara et al. 2022; Prabhakar and Rahimi Afzal 2019), ModelStar significantly extends its applicability. Unlike INNs, which focus primarily on abstracting fully connected networks for input perturbation analysis, ModelStar provides a general framework for analyzing weight perturbations across any parameterized linear layer (Section 4.1), offering a comprehensive solution for robustness verification.

The main contributions of our work include the following:

- We propose **ModelStar**, a reachability-based framework for verifying the robustness of neural networks against *weight perturbations*. ModelStar systematically integrates weight perturbations into reachable sets, specifically star sets, by leveraging the linearity of parameterized layers and perturbation matrices, enabling a rigorous analysis of neural network behavior under interval-based weight variations.
- ModelStar provides a unified framework for robustness verification against perturbations in any linear layer, including fully connected and convolutional layers. Unlike prior methods that impose structural constraints, ModelStar enables robustness verification for arbitrary subsets of perturbed parameters within a layer, enhancing its flexibility and applicability.
- We analyze how weight perturbations affect reachable sets while showing that ModelStar achieves an exact characterization of the output for a single perturbed layer and an over-approximate representation for multiple perturbed layers. Additionally, we establish complexity bounds on the size of the augmented reachable set providing insights into computational feasibility.
- Through numerical evaluations, we demonstrate that ModelStar establishes tighter robustness bounds than existing methods. ModelStar verifies up to 60% more samples in image classification tasks compared to previous approaches, highlighting its effectiveness in improving robustness certification for real-world applications.

The rest of this paper is organized as follows: Section 2 reviews related work on the significance of weight perturbations in NNs and existing approaches to robustness bounds against these perturbations. Section 3 outlines the problem setup, including reachability analysis and safety verification. Section 4 discusses our weight perturbation model as well as the formulation of ModelStar, detailing its methodology and theoretical analysis. Section 5 presents numerical experiments comparing ModelStar with existing approaches for verifying the robustness of fully connected NNs against continuous weight perturbations, while also evaluating its performance on convolutional NNs. Section 5.6 discusses the limitations of ModelStar and outlines potential directions for future work. Finally, Section 6 concludes the paper.

2 Related Work

In this section, we initially discuss background literature to highlight the adverse effects of perturbations in the weights of an NN. Subsequently, we explore previous work on verifying and quantifying the robustness of an NN against weight perturbations.

2.1 Impact of Weight Perturbations in NNs

Numerous studies in the literature have highlighted the sensitivity of NNs to weight perturbations, with many also proposing methods to mitigate this vulnerability.

Xiang et al. (2019) investigated the sensitivity of the outputs of a convolutional neural network (CNN) to weight perturbations, across all possible inputs. Their approach involves calculating the expected absolute variation in the CNN's output, given an input distribution. Sun et al. (2021) employed a gradient-based estimation over the loss function to identify the weight perturbations that cause the worst loss change. This work demonstrates that perturbations in even a few parameters can lead to a substantial decrease in prediction accuracy. Yan et al.

(2022) concentrated on the perturbations caused by device variations in Compute-in-Memory (CiM) Deep Neural Network (DNN) accelerator architectures. They demonstrated that even minor device variations can lead to large performance drops in the worst-case scenario, and they presented a gradient descent-based approach to search for this scenario.

Both Özdenizci and Legenstein (2022) and Echeberria-Barrio et al. (2022) highlight the potential for stealthily modifying NN operation for target samples with specific bit-flips, while preserving the performance of the NN for the rest of the input space. Yu et al. (2023) proved the existence of such weight perturbations and presented algorithms to determine them. Additional approaches to attack weights by flipping bits were proposed by Bai et al. (2023), and Qian et al. (2023) provided a survey of bit-flip attacks on DNNs and methods to defend against them. Multiple studies demonstrate the sensitivity of NNs to even single-bit flips (Gongye and Fei 2024; S. Li et al. 2024). In order to mitigate misclassification due to hardware faults, Maji et al. (2024) presented an NN accelerator with hardware-based fault detection, whereas Stutz et al. (2020) focused on reducing the impact of bit errors by injecting these errors during NN training.

Cheney et al. (2017), examined the robustness of NNs to random architecture and weight perturbations. All weights of a single layer were perturbed simultaneously, with the perturbations drawn from a Gaussian distribution centered around zero. Network performance was evaluated in terms of decrease in accuracy. Cheney et al. (2017) concluded that the network is more sensitive to perturbations in lower layers than in higher layers.

The aforementioned studies highlight the sensitivity of NNs to weight perturbations, necessitating formal robustness guarantees for their deployment in safety-critical systems. The next section discusses past contributions in this area.

2.2 Robustness Bounds against Weight Perturbations

Prior research presents methods to measure the robustness of an NN against weight perturbations. They quantify it in terms of the maximum ℓ_∞-norm bounded perturbations that may affect a set of weights without compromising the predictive performance of the NN. T.-W. Weng et al. (2020) applied this scheme to each row of weights in a fully-connected layer such that the classification output of the NN remains correct i.e., the output for the correct class remains higher than the output for any other class. They derived a lower bound on the maximum tolerable perturbation by utilizing linear upper and lower bound functions of the NN function. They discussed single-layer as well multi-layer perturbations for feedforward networks with ReLU activation function. They also discussed weight quantization while integrating the aforementioned bounds, and their results showed that this approach indeed improves accuracy compared to quantization without accounting for the certificated perturbation bounds.

Tsai et al. (2021a) also considered norm-bounded weight perturbations and presented a bound on the pair-wise class margin in their presence. Pair-wise class margin refers to the difference between two elements of the output of the NN's last fully-connected layer, with each element corresponding to one class. Again, the NN considered here is a feed-forward NN, though the activation function can be monotonic non-negative 1-Lipschitz. Tsai et al. (2021b) extended this work to joint input- and weight-perturbation and presented a bound on the error in the pair-wise class margin caused by the perturbations in this case. Essentially, T.-W. Weng et al. (2020) presented a bound on the disturbances being applied, whereas Tsai et al. (2021a,b) presented bounds on the NN output as a result of the disturbances. Tsai et al. (2021a,b) also presented loss functions to improve robustness.

The application of the weight perturbation bounds in the aforementioned approaches is limited to specific sets of weights. Tsai et al. (2021a) discussed a single bound on perturbations in an entire weight matrix, whereas T.-W. Weng et al. (2020) allowed each row of the weights/bias matrices in a fully-connected layer to have a distinct perturbation bound. On the other hand, ModelStar allows distinct perturbation bounds for arbitrary sets of parameters in a parameterized linear layer, offering greater flexibility of analysis. However, we perturb entire

weights tensors in our numerical evaluation for a fair comparison of the bounds presented by Tsai et al. (2021a) and T.-W. Weng et al. (2020) with ModelStar.

3 Preliminaries and Problem Setup

Next, we present notations and preliminary concepts for the ModelStar framework. Table 1 shows some of the mathematical notations used throughout the paper.

Table 1. Commonly used mathematical notation.

Notation	Description
$\mathbb{R}^n$	The set of real-valued vectors of dimension n.
$\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_k}$	The space of real-valued order k tensors with dimensions $n_1, n_2, \ldots, n_k$.
$e_k \in \mathbb{R}^n$	The k-th standard basis vector in $\mathbb{R}^n$, whose k-th entry is 1 and all others are 0.
$e_k \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_j}$	The k-th standard basis tensor in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_j}$, with a 1 at the k-th entry in its vectorized form and 0 otherwise.
$E_{\mathbb{R}^n}$	The set $\{e_1, e_2, \ldots, e_n\} \subset \mathcal{R}^n$ of all standard basis vectors of $\mathbb{R}^n$.
Singleton	A set with a single element.

3.1 Neural Network Architectures

We consider feed-forward neural networks for robustness verification and define a feed-forward neural network $\mathcal{N}$ as a sequence of layers $\mathcal{N} = \{L_i\}$, $i = 1, 2, \ldots, n$, where L_i denotes the function representative of the i-th layer's operation. In other words, if x is an input to $\mathcal{N}$, the output of the NN is

$$\mathcal{N}(x) = L_n(L_{n-1}(\ldots L_2(L_1(x)) \ldots))$$

Activation layers are indexed distinctly in our definition of $\mathcal{N}$. As our work extends the reachability-analysis based neural network verification (NNV) framework by integrating model perturbations, L_i can be the function corresponding to any layer supported by the NNV framework. As detailed in Lopez et al. (2023), the NNV framework supports a wide array of NN layers including fully-connected, convolutional, ReLU, max pooling, average pooling, batch normalization, and softmax layers, among others. This expands the applicability of ModelStar to model perturbation analysis of a wide variety of NNs. The scope of our framework allows the input x to $\mathcal{N}$ to be a tensor of any size compatible with the NN, but we limit the NN outputs considered in our work to $\mathcal{N}(x) \in \mathbb{R}^n$. It is possible, however, to generalize our framework to other output spaces provided that the resulting output reachable set is convex.

While $\mathcal{N}$ can contain any layers supported by NNV, our work focuses on the analysis of weight perturbations in *linear layers*, defined as follows:

Definition 3.1 (Linear Layer). *Let the i-th layer of an NN have two parameter tensors: a weight tensor W_i and a bias tensor b_i. Let the operation of this layer on an input tensor x be defined as:*

$$L_i(x) = W_i \odot x + b_i \tag{1}$$

where $\odot$ is any linear operation. Then, we refer to this layer as a linear layer. The input x, the weight W, the bias b and the output $L(x)$ can be tensors with any dimensions that conform to the operation defined in (1).

Definition 3.1 encompasses a variety of linear layers. When the layer under consideration is a fully-connected layer, $\odot$ represents matrix product, and when the convolutional layer is concerned, $\odot$ represents matrix convolution. We define the *weights range* of a linear layer as the absolute scalar difference between the maximum and minimum weights in the layer's weight tensor.

3.2 Reachable Sets

Our approach involves computing sets that encapsulate all possible outputs of a layer as a result of parameter perturbations. These sets are then propagated through the remaining layers of the NN, with safety decisions based on the final output set. The central component of this approach is the concept of *reachable set*, defined as follows:

Definition 3.2 (Reachable Set). *Given a set $\mathcal{I}$ as an input to the network $\mathcal{N}$, the output reachable set $\mathcal{R}_i$ of the i-th layer is obtained by applying the layer's function L_i to the output reachable set of the preceding layer (Tran, Bak, et al. 2020a).*

$$\mathcal{R}_i = \{y \mid y = L_i(x),\ x \in \mathcal{R}_{i-1}\}$$

The input set to the first layer, $\mathcal{R}_0$, is the input $\mathcal{I}$ to the NN, and the output set of the last layer, $\mathcal{R}_n$, is the reachable set of the NN.

ModelStar propagates a modified star set through NN layers (Tran, Bak, et al. 2020a) with perturbed weights (as detailed in Section 4), producing an output set that encompasses all possible outcomes resulting from the specified weight perturbations.When L_i is a non-linear function, the ModelStar framework employs an over-approximate reachable set $\mathcal{R}'_i$ ($\supseteq \mathcal{R}_i$) for scalability (Tran, Bak, et al. 2020a). Next, we define the *star set* structure used to represent the reachable set in ModelStar.

3.3 Star Sets

To generalize our results and simplify exposition, we define star sets differently than previous works (Tran, Bak, et al. 2020a; Tran, Manzanas Lopez, et al. 2019):

Definition 3.3 (Star Set). *A star set $\mathcal{I}$ is defined by a central point c, and m basis tensors v_j, $j = 1, \ldots, m$, that span a space around the origin c. The star set is represented by the tuple $\langle c, V, P\rangle$ where V is the set of m basis tensors $\{v_1, v_2, \ldots, v_m\}$, and P is a set of predicates, limited to linear constraints. The set is mathematically defined as follows:*

$$\mathcal{I} = \{x \mid x = c + \sum_{j=1}^{m} \alpha_j v_j,\ C\alpha \leq d\} \tag{2}$$

*where $x, c, v_j \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_k}$, $j = 1, \ldots, m$, and α is a vector of m predicate variables $[\alpha_1, \alpha_2, \ldots, \alpha_m]$. If $P(\alpha) \triangleq C\alpha \leq d$ contains p linear constraints, then $C \in \mathbb{R}^{p \times m}$ and $d \in \mathbb{R}^{p \times 1}$. Each pair (α_j, v_j) is referred to as a **generator**. Hence, the set in (2) comprises m generators.*

We have presented the general form of star set as consisting of tensors of arbitrary dimensions $n_1, n_2, \ldots, n_k$ in order to state our theorems across various layer sizes. However, the simplest form of star set, with $x, c, v_j \in \mathbb{R}^n$, has been utilized by Tran, Manzanas Lopez, et al. (2019). Later, Tran, Bak, et al. (2020a) extended the concept of star set to ImageStar set whose elements are images with multiple channels i.e., $x, c, v_j \in \mathbb{R}^{h \times w \times n_c}$, where h, w and n_c are the height, width and number of channels of the image, respectively. We note that the general star set in (2) can always be flattened so that $x, c, v_j \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_k}$ are represented as vectors $x', c', v'_j \in \mathbb{R}^n$ with $n = n_1 \times n_2 \times \cdots \times n_k$. This is convenient for analyses that depend only on the entries of x and not on its tensor shape (for example, computing element-wise upper and lower bounds for x over the set). Tran, Bak, et al. (2020a) mapped perturbations in the NN's input to the star set's generators and discussed the propagation of the

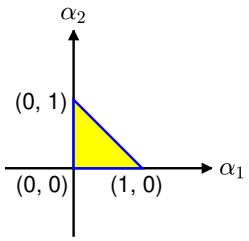

Fig. 2. Graphical depiction of the star set defined by the parameters

$$c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, P \equiv \begin{pmatrix} 0 \leq \alpha_1 \leq 1, \\ 0 \leq \alpha_2 \leq 1, \\ \alpha_1 + \alpha_2 \leq 1 \end{pmatrix}$$

star sets through various NN layers, such as fully-connected, convolutional, ReLU, batch normalization, average pooling and max pooling layers. *Our contribution extends this framework for the analysis of weight perturbations by mapping these perturbations to the generators while maintaining the star set structure.* This approach allows us to leverage reachability analysis methods for verifying NNs against weight perturbations.

Next, we discuss some interesting properties of star sets that motivate their use for NN verification through reachability analysis (Tran, Bak, et al. 2020a):

- Star sets are convex sets. This allows us to verify whether the output reachable set intersects with a given convex unsafe region using a convex program. The convexity of star sets follows from the fact that they encode linear transformations of bounded polytopes. To observe this, consider a flattened star set with $x, c, v_j \in \mathbb{R}^n$. This star set has two main components: a bounded polytope in the α-space defined by $C\alpha \leq d$, and a linear transformation of this α-space to the x-space:

  $$x = c + V\alpha$$

  where α is the vector $[\alpha_1, \alpha_2, \ldots, \alpha_m]^T$ and V is the transformation matrix with column vectors consisting of v_i's:

  $$V = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}$$

  Figure 2 illustrates an example star set in the α-space.
- Star sets can be conveniently propagated through linear layers by applying the layer's operation on the generators of the star set. Given a linear layer L_i (Definition 3.1) and a star set $\mathcal{I}$ (Definition 3.3) as input to the layer, the reachable set of this layer is another star set

  $$\mathcal{R}_i = L_i(\mathcal{I}) = \{x \mid x = W_i \odot c + b + \sum_{j=1}^{m} \alpha_j W_i \odot v_j,\ C\alpha \leq d\}$$

  $$\mathcal{R}_i = \{x \mid x = c' + \sum_{j=1}^{m} \alpha_j v'_j,\ C\alpha \leq d\}$$

  where $c' = W_i \odot c + b$ and $v'_j = W_i \odot v_j$.
- Star sets can be propagated through a number of non-linear layers (such as ReLU and max pooling layers) through addition and manipulation of generators and constraints. In particular, in our work, we are able to encode weight perturbations as generators and constraints that are added to the star set input to the perturbed layer.

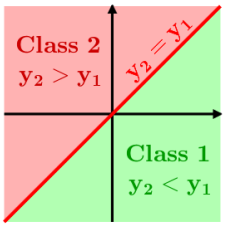

Fig. 3. Safe and unsafe sets for the outputs of a binary classifier *when the input belongs to class 1*. The safe set, $y_2 < y_1$, is shaded in green, whereas the unsafe set, $y_2 \geq y_1$, is shaded in red.

3.4 Safety Verification

In supervised learning, a neural network (NN) is trained to carry out a well-defined task, such as classification or regression. Each task is accompanied by explicit correctness requirements on the network outputs, typically expressed as inequalities or logical constraints that encode desired behaviors such as correct class ranking, bounded prediction error, or adherence to safety margins. The subset of the NN's output space that violates these requirements is referred to as the *unsafe set*, while its complement is called the *safe set*. The precise definition of the unsafe set is task dependent and forms the basis of safety and robustness verification.

In this paper, we consider image classification, airborne collision avoidance, and aircraft component remaining useful life (RUL) prediction tasks to evaluate the performance of our robustness verification approach. We choose these tasks because they are among the commonly used benchmarks in the NN verification literature (Bak and Tran 2022; Brix et al. 2024; Katz et al. 2017; Tran, Bak, et al. 2020a). To give an intuitive example of an unsafe set, we formulate the unsafe set for the classification problem below.

Let the output of a classifier with n classes be represented by a vector $y \in \mathbb{R}^n$, where y_i denotes the predicted score or probability associated with class i. If the true class is k, then any output for which the score of an incorrect class equals or exceeds that of the correct class constitutes a misclassification. Accordingly, for each incorrect class $i \neq k$, we define a half-space specified by the inequality $y_i \geq y_k$. The unsafe set is then given by the union of these half-spaces, while the safe set corresponds to their complement. Formally, the safe and unsafe output sets are defined as:

$$Y_{\text{safe}} = \bigcap_{\substack{i=1 \\ i \neq k}}^{n} \{y \mid y_k > y_i\},$$

$$Y_{\text{unsafe}} = \bigcup_{\substack{i=1 \\ i \neq k}}^{n} \{y \mid y_i \geq y_k\}.$$

Figure 3 illustrates the safe and unsafe regions for the output of a binary classifier when the input belongs to class 1. More generally, for regression tasks, the unsafe set can be defined in terms of task-specific performance criteria, such as predictions that lie outside a prescribed tolerance interval around a reference value.

Besides the unsafe set, the second major component of safety verification is the reachable set produced by the NN under perturbations. The reachable set $\mathcal{R}_n$ of a NN with n layers captures all outputs attainable by the NN under the considered uncertainties. Safety is then determined by checking whether this reachable set intersects the unsafe set. If the intersection is empty, the NN is certified to satisfy the specified safety requirements for all admissible perturbations. On the other hand, a non-empty intersection indicates that the NN can violate the

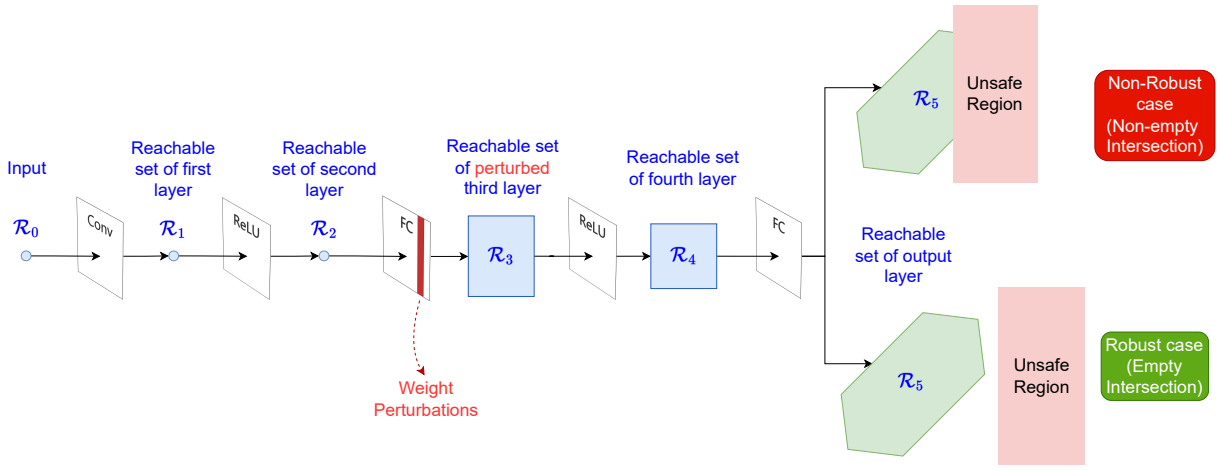

Fig. 4. Illustration of reachable set propagation through the NN shown in Fig. 1. The terms 'Conv' and 'FC' stand for convolutional layer and fully-connected layer, respectively.

specification under admissible perturbations. Figure 4 illustrates these scenarios along with the propagation of reachable sets through a representative NN. The NN input is a singleton, and so are the reachable sets of all layers up to the perturbed fully-connected layer. The perturbation at this layer increases the size of the reachable set, which is then propagated through the subsequent layers. For safety verification, only the reachable set at the output layer is required: if it intersects the unsafe set, the NN is not robust to the considered perturbations; if the intersection is empty, the NN is robust. Both cases are illustrated in Figure 4. Since the softmax layer is monotone, propagating reachable sets through it is unnecessary for verifying robustness with respect to misclassification.

Basically, given an n-layer NN subject to perturbations and an unsafe output set Y_{unsafe}, the goal is to compute the output reachable set $\mathcal{R}_n$ of the NN which allows safety to be verified by checking whether

$$\mathcal{R}_n \cap Y_{\text{unsafe}} = \emptyset.$$

However, computation of $\mathcal{R}_n$ can be computationally intractable. As such, our work focuses on computing an over-approximation $\mathcal{R}_n'' \supset \mathcal{R}_n$ such that $\mathcal{R}_n'' \cap Y_{\text{unsafe}} = \emptyset$. When this condition cannot be satisfied (i.e., $\mathcal{R}_n'' \cap Y_{\text{unsafe}} \neq \emptyset$), ModelStar cannot conclusively certify robustness of the NN against the specified perturbations. Despite this conservatism inherent to over-approximation methods, our experimental evaluation in Section 5 demonstrates that ModelStar successfully certifies robustness against a significantly larger number of perturbation scenarios than existing approaches.

When the unsafe set Y_{unsafe} is non-convex, we represent it as a finite union of convex unsafe sets to enable tractable verification within our framework. Safety verification can then be carried out by solving a separate convex optimization problem to check intersection with each convex unsafe set (Tran, Bak, et al. 2020a). In the applications considered in this work, however, unsafe sets are defined exclusively using half-spaces, which allows safety verification to be performed using linear programs (LPs).

We refer to the degree of over-approximation present in a reachable set as *conservativeness*. As a reachable set propagates through an NN, any operation that introduces over-approximation in the reachable set increases the likelihood of a non-empty intersection between the unsafe set and the over-approximate output reachable set of the NN. Such intersections can occur even if the actual output reachable set of the NN does not intersect with the unsafe set. Increase in the likelihood of this intersection is referred to as an increase in over-approximation, or an

increase in conservativeness. For instance, in the context of image classification, we assess the robustness of an NN by counting the number of input samples for which the output reachable set does not intersect with any unsafe half-plane. Consequently, as over-approximation increases, the number of images for which ModelStar can verify the NN's robustness-termed *verifiable robustness*-tends to decrease.

4 ModelStar: Reachability Analysis Framework for Model Perturbations

We begin this section by presenting our perturbation model. Next, we discuss how weight perturbations in a linear layer of an NN are translated to the generators in the star reachable set, under the condition that there are no perturbations prior to the layer under consideration. Lastly, we present the general case where perturbations already exist in the star set input to the perturbed linear layer.

The output of the linear layer discussed in this section is the pre-activation output. Similarly, the output reachable set is the pre-activation output reachable set; we rely on the existing implementation of ImageStar (Tran, Bak, et al. 2020a) to propagate this set through the activation layer.

4.1 Weight Perturbation Model

In an NN, perturbations in the i-th layer can change the outputs of that layer, forming a reachable set defined as:

$$\mathcal{R}'_i = \{y \mid y = L'_i(x),\ x \in \mathcal{R}_{i-1}\}$$

Here, $L'_i(x)$ denotes the perturbed function of the i-th layer, and $\mathcal{R}'_i$ is the resulting reachable set (see Definition 3.2). Although non-linear layers can experience perturbations, our work focuses on perturbations in the parameters of linear layers only (Definition 3.1). Specifically, our experimental evaluation (Section 5) is based on perturbations in fully connected and convolutional layers, but it is straightforward to implement the results in this section for other linear layers.

The perturbations considered in our paper are scalar, bounded perturbations. To encode the locations of the parameters subject to a perturbation, we present the concept of *perturbation map*:

Definition 4.1 (Perturbation Map). *Let the i-th layer of an NN be a linear layer with weight and bias tensors W_i and b_i, and consider m interval-bound perturbations $\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m\}$, each with its own lower and upper bounds, affecting W_i and b_i. The weight perturbation map corresponding to ε_j, denoted by M_{Wj}, is defined as a tensor with the same dimensions as W_i, where each entry is non-zero if the corresponding weight is affected by ε_j, and zero otherwise. Similarly, the bias perturbation map corresponding to ε_j, denoted by M_{bj}, is defined as a tensor with the same dimensions as b_i, where each entry is non-zero if the corresponding bias is affected by ε_j, and zero otherwise.*

$$W' = \underbrace{\begin{bmatrix} 2 & 1 \\ 3 & 5 \end{bmatrix}}_{W} + \varepsilon_1 \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}}_{M_{W1}} + \varepsilon_2 \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0.5 \end{bmatrix}}_{M_{W2}}$$

$$b' = \underbrace{\begin{bmatrix} -2 \\ 1 \end{bmatrix}}_{b} + \varepsilon_1 \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{M_{b1}} + \varepsilon_2 \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{M_{b2}}$$

Fig. 5. A 2×2 weight matrix W and a 2×1 bias matrix b are subject to two bounded perturbations with magnitudes ε_1 and ε_2, resulting in the perturbed weight matrix $W' = W + \varepsilon_1 M_{W1} + \varepsilon_2 M_{W2}$ and perturbed bias matrix $b' = b + \varepsilon_1 M_{b1} + \varepsilon_2 M_{b2}$. ε_1 affects the weight at location $(2, 2)$ whereas ε_2 affects the weights at locations $(2, 1)$ and $(2, 2)$ and the bias at location $(2, 1)$.

Figure 5 shows a 2×2 weight matrix and a 2×1 bias matrix subject to two perturbations ε_1 and ε_2. This example illustrates how the perturbation map M_{W2} represents the effect of the perturbation ε_2 on two weights simultaneously. While this effect could also be represented by two separate perturbations and maps—one perturbation being a scaled version of the other—a single perturbation map can effectively represent multiple linearly dependent perturbations, reducing dimensionality and simplifying the analysis. However, perturbations that are not linearly dependent must be treated the same as independent perturbations, i.e., each of these perturbations requires its own scalar perturbation variable and corresponding perturbation map.

4.1.1 Application to Floating-Point Errors: As an illustrative example of modeling weight error bounds via perturbation maps, we consider *absolute floating-point errors* that may vary across the scalar elements of a tensor $W \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q}$ stored in memory (Izycheva and Darulova 2017). The tensor W contains $m = \prod_{j=1}^{k} n_j$ scalar elements. We can model the error on the k-th scalar element of W (under a fixed flattening of the tensor indices) by a perturbation variable ε_k, which is bounded by an element-wise, absolute tolerance ϵ_k derived from standard floating-point error bounds (Goldberg 1991):

$$-\epsilon_k \leq \varepsilon_k \leq \epsilon_k, \quad k = 1, \ldots, m.$$

These perturbation variables are then accompanied by m perturbation maps:

$$\{e_1, e_2, \ldots, e_m\} \subset \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q}$$

Each map $e_k = M_{Wk}$ corresponds to the perturbation ε_k applied to the k-th element of W. In our experimental evaluation in Section 5, we use this setting of layer-wide, independent, element-wise perturbations for comparisons with existing robustness bounds (Tsai et al. 2021a; T.-W. Weng et al. 2020). However, since these existing methods do not support distinct perturbation bounds for individual weights, for a fair comparison, we subject all weights in each layer to perturbations with the same lower and upper bounds in our experiments.

In this subsection, we introduced our perturbation modeling technique with examples. Next, we employ perturbation maps to construct the perturbed layer's reachable set $\mathcal{R}'_i$ in two distinct scenarios: single perturbed layer, and multiple perturbed layers. Although fully-connected and convolutional layers may have inputs and outputs of varying sizes, our analysis exploits the inherent linearity of these layers in order to apply the layer's operation to the weight perturbation maps and the input. The weight perturbation maps M_{Wj} are defined to have the same size as the weight tensor W, ensuring that their interaction with the input—either through convolution or multiplication—is consistent with the original layer operation. Consequently, the resulting basis tensors v_j (as derived in subsequent sections) have the same size as the nominal output tensor of the layer. In this manner, the assumptions regarding linearity and perturbation map size maintain dimensional consistency throughout our analysis and eliminate the need to specify tensor dimensions for each case explicitly.

4.2 Reachable Set for First Perturbed Layer

The following theorem constructs the reachable set for the scenario where the input to the linear layer under consideration is a singleton. This scenario occurs when there are no perturbations in any layer before the layer under consideration, and the input to the NN is also a singleton.

Theorem 4.1. *Let the i-th layer of an NN be a linear layer with unperturbed layer operation $L_i(x) = W_i \odot x + b_i$, and let the input to the layer be the singleton $\{x\}$. If there are a total of m independent perturbations $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ affecting the tensors W_i and b_i, the output reachable set of this layer is represented exactly by a star set comprising m generators.*

Proof. Consider the vector of perturbations $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]$ constrained by the lower and upper bounds α_l and α_u, respectively. Each perturbation α_j has an associated weight perturbation map M_{Wj} and an associated bias perturbation map M_{bj}. Let us define the star set $\langle c, V, P \rangle$ using the central point $c = W_i \odot x + b_i$, the set V of

m basis tensors $\{v_1, v_2, \ldots, v_m\}$ such that $v_j = M_{Wj} \odot x + M_{bj}$, and the predicate set $P(\alpha) = \underline{\alpha} < \alpha < \overline{\alpha}$. We will now demonstrate that this star set is an exact representation of the reachable set of the perturbed linear layer under consideration.

The perturbed weight and bias tensors W_i' and b_i' can be expressed as:

$$W_i' = W_i + \sum_{j=1}^{m} \alpha_j M_{Wj}, \; b_i' = b_i + \sum_{j=1}^{m} \alpha_j M_{bj}$$

When the layer's perturbed operation, $L_i'(x) = W_i' \odot x + b_i'$, is applied to the input x, the output is

$$\begin{aligned} L_i'(x) &= W_i \odot x + b_i + \sum_{j=1}^{m} \alpha_j M_{Wj} \odot x + \sum_{j=1}^{m} \alpha_j M_{bj} \\ &= c + \sum_{j=1}^{m} \alpha_j \left(M_{Wj} \odot x + M_{bj} \right) \\ &= c + \sum_{j=1}^{m} \alpha_j v_j \end{aligned}$$

Hence, the output reachable set of the perturbed linear layer can be represented exactly by the following star set with m generators (α_j, v_j):

$$\mathcal{R}_i' = \{ y \mid y = c + \sum_{j=1}^{m} \alpha_j v_j, \; \underline{\alpha} \leq \alpha \leq \overline{\alpha} \} \tag{3}$$

□

Essentially, given a perturbation α_j and the locations of the weights and biases affected by α_j, Theorem 4.1 encompasses the effect of the perturbation α_j on the output neurons of the linear layer by using the basis tensor v_j. In doing so, Theorem 4.1 provides information about the exact structure of the output reachable set that results from the perturbations in the first perturbed linear layer. The structure of the star set allows for the representation of this reachable set without any conservativeness, leading to tighter bounds on the robustness of the NN, as demonstrated in Section 5.

As Theorem 4.1 states, each perturbation introduces a new generator (α_j, v_j). Whenever a star set with a basis vector v_j is propagated through a subsequent k-th linear layer, each basis vector v_j in the star set is transformed to a new basis vector $v_j' = W_k \odot v_j$ (Tran, Bak, et al. 2020a), where $\odot$ represents the linear operation associated with the k-th layer. Hence, the resulting basis vector v_j' has the same size as the output of the k-th layer. Propagation through certain non-linear layers, such as ReLU and max pooling, requires computing element-wise bounds of the star set, which in turn entails solving a large number of LPs (Tran, Bak, et al. 2020b). The number of such LPs typically grows with the number of neurons. Therefore, after the star set has passed through linear layers with many neurons, the increased dimensionality of the basis vectors can lead to substantial memory consumption and longer execution times in subsequent non-linear layers. However, the dimensionality of each of these LPs scales with the number of generators. Thus, the resource usage of these LPs can be reduced by employing over-approximations that decrease the number of generators, which motivates our next result.

Theorem 4.1 provides an exact representation $\mathcal{R}_i$ of the reachable set, but this representation contains m generators, where m is the number of independent perturbations. Let n denote the number of neurons in the linear layer under consideration in Theorem 4.1. Algorithm 1 presents a method to construct a box-like over-approximation of the exact reachable set $\mathcal{R}_i$ that reduces the number of generators from m to (at most) n, at the cost of potential additional conservativeness. This reduction is particularly beneficial when $m \gg n$, since in such

Algorithm 1: Generator–transformation for bounding box over-approximation of the reachable set in Theorem 4.1

Input: Generator set $V = \{v_j\} \subset \mathbb{R}^n$, $j = 1, \ldots, m$, and predicate variable bounds $\underline{\alpha}, \overline{\alpha} \in \mathbb{R}^m$ of flattened star set $\mathcal{R}'_i = \{x \mid x = c + \sum_{j=1}^{m} \alpha_j v_j,\ \underline{\alpha} \le \alpha \le \overline{\alpha}\}$.

Output: Generator set $V' \subseteq E_{\mathbb{R}^n}$ with $|V'| = q \le n$ and predicate variable bounds $\underline{\alpha'}, \overline{\alpha'} \in \mathbb{R}^q$ of flattened star set $\mathcal{R}''_i = \{x \mid x = c + \sum_{k=1}^{q} \alpha_k v'_k,\ \underline{\alpha'} \le \alpha \le \overline{\alpha'}\}$.

Initialize $\ell, u \in \mathbb{R}^n$ to 0 // Bounds of bounding box
for $k \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** m **do**
    $z_{jk} \leftarrow v_{j(k)}$ // k-th element of vector v_j
    $\ell_k \leftarrow \ell_k + \min(\underline{\alpha}_j z_{jk}, \overline{\alpha} z_{jk})$
    $u_k \leftarrow u_k + \max(\underline{\alpha}_j z_{jk}, \overline{\alpha} z_{jk})$
  end
end
$V' \leftarrow \emptyset$
$p \leftarrow 1$
$q \leftarrow$ **count_nonzero_elements**$(u - \ell)$ // Number of new predicate variables
Initialize $\underline{\alpha'}, \overline{\alpha'} \in \mathbb{R}^q$ to 0
for $k \leftarrow 1$ **to** n **do**
  if $u_k - l_k > 0$ **then**
    $V' \leftarrow V' \cup \{e_k\}$
    $\underline{\alpha'}_p \leftarrow \ell_k, \quad \overline{\alpha'}_p \leftarrow u_k$
    $p \leftarrow p + 1$
  end
end
return $V', \underline{\alpha'}, \overline{\alpha'}$

cases storing a star set with m generators may be practically infeasible, and the reduced-generator representation accelerates propagation through subsequent non-linear layers. In the following theorem, we prove that the output of Algorithm 1, which is a bounding box in the form of a star set $\mathcal{R}''_i$, is a sound over-approximation of $\mathcal{R}'_i$, and also discuss a case in which $\mathcal{R}''_i$ is exactly equivalent to $\mathcal{R}'_i$. While Algorithm 1 is presented for the flattened form of $\mathcal{R}_i$ for notational convenience, the algorithm is the same for the general $\mathcal{R}_i$ with any tensor dimensions, as shown by the following theorem.

Theorem 4.2. *Let the parameters of a linear layer with n neurons be subject to m independent perturbations. This layer's exact reachable set, $\mathcal{R}'_i$, that has m generators (Theorem 4.1), is over-approximated by a star set $\mathcal{R}''_i = \langle c, V', P' \rangle$ that encodes a bounding box and has (at most) n generators. However, if each basis tensor v_j in $\mathcal{R}'_i$ has at most one non-zero entry, then $\mathcal{R}'_i = \mathcal{R}''_i$, i.e., the reachable set is represented **exactly** by a star set with (at most) n generators in this case.*

Proof. We accomplish the proof of Theorem 4.2 through a particular translation of the basis $\{v_1, v_2, \ldots, v_m\}$ of the set $\mathcal{R}'_i$ presented in Theorem 4.1. Since the linear layer under consideration has n neurons, each basis tensor

v_j in $\mathcal{R}'_i$ can be split into (at most) n tensors v_{jk}, $k = 1, \ldots, n$, with each v_{jk} containing one of the non-zero entries of v_j, as follows:

$$v_j = v_{j1} + v_{j2} + \cdots + v_{jn}$$

Thus, the product $\alpha_j v_j$ present in $\mathcal{R}'_i$ can be expressed as:

$$\alpha_j v_j = \alpha_j v_{j1} + \alpha_j v_{j2} + \cdots + \alpha_j v_{jn} \tag{4}$$

If v_j has any zero entries, the corresponding v_{jk} will be zero. However, for the sake of simplicity, we have listed the v_{jk} for all $j = 1, \ldots, m$ and $k = 1, \ldots, n$ in our discussion. We denote the sole non-zero scalar element of each non-zero v_{jk} by z_{jk}. For each non-zero v_{jk}, the division v_{jk}/z_{jk} results in a tensor whose only non-zero entry is 1. We denote this tensor by e_{jk}. Thus, we can express $\alpha_j v_j$ as follows:

$$\begin{aligned} \alpha_j v_j &= \alpha_j z_{j1} e_{j1} + \alpha_j z_{j2} e_{j2} + \cdots + \alpha_j z_{jn} e_{jn} \\ \alpha_j v_j &= \alpha_{j1} e_{j1} + \alpha_{j2} e_{j2} + \cdots + \alpha_{jn} e_{jn} \end{aligned} \tag{5}$$

where $\alpha_{jk} = \alpha_j z_{jk}$. By splitting all m generators of $\mathcal{R}'_i$ as discussed above, we obtain (at most) mn terms of the form $\alpha_{jk} e_{jk}\ \forall\ j = 1, \ldots, m\ \wedge\ k = 1, \ldots, n$. Since $e_k = e_{1k} = e_{2k} = \cdots = e_{mk}\ \forall\ k = 1, \ldots, n$, we can combine the predicate variables accompanying the various instances of e_k into a single predicate variable as follows:

$$\alpha'_k = \alpha_{1k} + \alpha_{2k} + \cdots + \alpha_{mk}$$

The upper and lower bounds of α'_k are obtained by adding together the upper and lower bounds of the variables $\alpha_{jk}\ \forall\ j = 1, \ldots, m$ (lines 1-8 of Algorithm 1). Thus, we can over-approximate $\mathcal{R}'_i$ by the following set:

$$\mathcal{R}''_i = \{y \mid y = c + \sum_{k=1}^{n} \alpha'_k e_k,\ \underline{\alpha'} \leq \alpha' \leq \overline{\alpha'}\} \tag{6}$$

where α' is the vector of predicate variables $[\alpha'_1, \alpha'_2, \ldots, \alpha'_n]$, and $\underline{\alpha'}$ and $\overline{\alpha'}$ are the lower and upper bounds of α', respectively.

Next, we discuss the cause of the over-approximation in $\mathcal{R}''_i$. Although the α_{jk} variables are scaled versions of α_j, we have treated the α_{jk} as independent variables in order to translate the star set's bases from v_j to the standard basis tensors e_k for reducing set size. As $k = 1, \ldots, n$, each α_j is split into (at most) n variables (we say *at most* because there are no variables α_{jk} corresponding to zero elements of v_j). This treatment artificially inflates the dimensionality of the α-space from m to (at most) mn. In other words,

$$\text{span}\{v_1, \ldots, v_m\} \subseteq \text{span}\{v_{jk}\ \forall\ j, k\} \subseteq \text{span}\{e_1, \ldots, e_n\}$$

If $v_{jk} = 0\ \forall\ j = 1, \ldots, m$, the generator corresponding to e_k is excluded from the set $\mathcal{R}''_i$, i.e., Algorithm 1 does not append such a generator to the new set of generators (see lines 9-19 of Algorithm 1). However, if this is not the case for any k, then in this special case,

$$\text{span}\{v_{jk}\ \forall\ j, k\} = \text{span}\{e_1, \ldots, e_n\}$$

Lastly, we discuss the special case where there is no over-approximation. Since splitting is only required for each of the non-zero entries in v_j, the dimensionality actually only inflates to the total number of non-zero entries in all v_j. Thus, no splitting is required and no over-approximation is introduced during reduction of set size if each v_j, $j = 1, \ldots, m$, has at most one non-zero element. In this case,

$$\text{span}\{v_1, v_2, \ldots, v_m\} \subseteq \text{span}\{e_1, e_2, \ldots, e_n\} \tag{7}$$

As with the general case, (7) holds with equality if there exists some $v_{jk} \neq 0\ \forall\ j = 1, \ldots, m$.

□

Algorithm 1 enables cheaper construction and propagation of a reachable set that can account for perturbations in large layers containing millions of weights, provided the number of neurons in the layer is significantly smaller than the number of weights. The over-approximation introduced in this process results from artificially inflating the dimensionality of the set, but this trade-off reduces the number of generators, balancing computational efficiency with a corresponding loss of information. However, we derive the following corollary from Theorem 4.2 to bound the size of the reachable set of a perturbed fully-connected layer without any over-approximation for a special case, as follows:

Corollary 4.1. *Consider a fully-connected layer with weight matrix $W \in \mathbb{R}^{n\times p}$ and bias vector $b \in \mathbb{R}^{n\times 1}$. Let W and b be affected by m independent perturbations such that each perturbation affects at most one row of W, though different perturbations can affect different rows of W. If α_j affects row k of W, it may only affect the k-th element of b, and vice versa. Then, the output reachable set of this layer is represented exactly by a star set with n generators.*

Proof. According to the corollary statement, a perturbation α_j must have perturbation maps M_{Wj} and M_{bj}, for the weight and bias matrices respectively, such that if the k-th row of M_{Wj} has any non-zero elements, only the k-th row of M_{bj} can have non-zero elements, and vice versa. As only the k-th rows of M_{Wj} and M_{bj} may be non-zero, the resulting basis vector $v_j = M_{Wj}x + M_{bj}$, where $x \in \mathbb{R}^{p\times 1}$, can only have one non-zero element, i.e., its k-th element. Thus, as discussed at the end of Theorem 4.2, no splitting of generators is required for reduction of set size, and the remaining process for translation of basis is the same as that followed in the proof of Theorem 4.2 after the splitting of generators, i.e., after (4). Since no splitting is performed, no over-approximation is introduced in this case. □

Corollary 4.1 allows *exact* representation of the effects of perturbing a fully-connected layer for any $m > n$ using only n generators instead of the m generators required by Theorem 4.1, so long as no perturbation spans more than one row of the parameter matrices.

In this section, we presented the construction of the reachable set when the input to the perturbed linear layer is a singleton. Next, we extend this discussion to the general case, where the input to a layer is not limited to a single element.

4.3 Reachable Set for Subsequent Perturbed Layers

Now we consider a perturbed linear layer present after a previous perturbed layer in the NN. The input to the current layer is the reachable set of the previous layer, i.e., a star set containing predicate variables that encode perturbations. Perturbations in the current layer are also represented using predicate variables. The linear operation of the current layer results in products between predicate variables from the input set and those in the current layer's parameters, leading to non-linear behavior. To address this, ModelStar adopts an over-approximation approach to maintain the linearity of the reachable set.

Theorem 4.3. *Let the i-th layer of an NN be a linear layer with unperturbed layer operation $L_i(x) = W_i \odot x + b_i$. Let there be q independent perturbations affecting W_i and b_i. If the input to the layer is a star set $\mathcal{I}$ with m generators and the layer has n neurons, then the output reachable set of this layer is over-approximated by a star set comprising $m + n + q$ generators.*

The proof of Theorem 4.3, provided in Appendix A, shows that the m predicate variables from the input set interact with the q predicate variables from the perturbations in the current layer, producing mq new generators in the reachable set of the current layer. The proof also discusses how Algorithm 1 can be applied to combine these mq generators into $\boldsymbol{n}$ **generators** at the cost of over-approximation of the reachable set. Finally, the perturbations in the bias tensor are handled in the same fashion as Theorem 4.1, resulting in $\boldsymbol{q}$ **generators**. Hence, despite the hard-to-tame non-linear behavior, the proof of Theorem 4.3 offers a memory-friendly reachable set representation

adding at most $n + q$ **generators** to the reachable set being propagated through the perturbed linear layer. However, this comes at the cost of higher conservativeness compared to the single perturbed layer scenario, as evidenced by the experiments in Section 5.

In this section, we have presented the core structure of ModelStar. The key features of our approach include representation of weight perturbations using generators in the star set, and propagation of the star set through a perturbed layer. In the next section, we compare the performance of ModelStar with other state-of-the-art approaches.

5 Evaluation of ModelStar

In this section, we evaluate ModelStar over five NNs: an MLP trained on the MNIST dataset, a CNN trained on the MNIST dataset, the VGG19 CNN trained on the ImageNet dataset, an MLP from the ACAS Xu set of MLPs and a CNN for remaining useful life (RUL) prediction of aircraft components[1]. We choose these architectures and datasets due to their use in NN safety research (Bak and Tran 2022; Kirov and Rollini 2023; Kundu et al. 2021; Tran, Bak, et al. 2020b; Xu et al. 2023). In our experiments, we introduce varying amounts of perturbation in the weights of fully-connected and convolutional layers of the NNs. We present the magnitude of perturbation affecting a layer relative to the weights range of the layer. This is motivated by the fact that such a relation between the perturbation magnitude and the range of weights translates directly to practical applications such as floating-point errors as well as finding the minimum tolerance threshold for programming weights to memristors in CiM devices using write-verify (Yan et al. 2022). Unlike ModelStar, the weight-perturbation models in state-of-the-art approaches do not support (i) analysis of the impact of perturbations on individual weights, or (ii) distinct perturbation bounds for the perturbations affecting each weight (Tsai et al. 2021a; T.-W. Weng et al. 2020). Thus, for fair comparison with state-of-the-art approaches, in subsequent experiments, unless specified otherwise, **we perturb all weights in one layer of an NN at a time, one perturbation affecting each weight, with all perturbations having identical bounds**. We represent the layer-wide perturbation by the ∞-norm bound on the perturbation magnitude, and display this bound in terms of the percentage of the affected layer's weights range. All experiments have been conducted in MATLAB running on a 64-core Ubuntu machine with 512GB RAM.

For image classification tasks, we consider a set of input images that are correctly classified by the NN in the absence of perturbations. Thus, NN robustness is quantified by counting the number of inputs safely classified by the NN despite the presence of weight perturbations. It is important to note that even when Theorem 4.1 is used to construct an exact representation of the reachable set of the perturbed layer, the output reachable set of the NN is still over-approximate because the propagation of reachable sets through non-linear layers introduces over-approximation (Tran, Bak, et al. 2020a). Hence, the *verifiable robustness* (Section 3.4) of an NN, obtained through ModelStar, is either the same as, or less than, the actual robustness of the NN. However, the experiments demonstrate that ModelStar is still able to verify the robustness of NNs against weight perturbations for a larger number of cases compared to state-of-the-art approaches (Tsai et al. 2021a; T.-W. Weng et al. 2020).

5.1 Multi-Layer Perceptron (MNIST)

MNIST is a dataset containing 28×28 images of hand-written digits with 10 classes-one for each digit from 0 to 9. The MLP under consideration has been trained on the MNIST dataset with an accuracy of 99.72%. It has 5 hidden fully-connected layers with 1024 and 512 neurons in the first two layers respectively, and 256 neurons in each of the remaining three. Each fully-connected hidden layer is followed by a ReLU layer. We attempt to verify the robustness of the NN for 200 input images, 20 per class of the MNIST dataset. Verification results for single-layer perturbation are presented first, followed by results for multi-layer perturbation.

[1]The code is available on both Zenodo (Zubair, T. Johnson, et al. 2026) and GitHub (https://github.com/cirens-utd/nnv_weight_perturb).

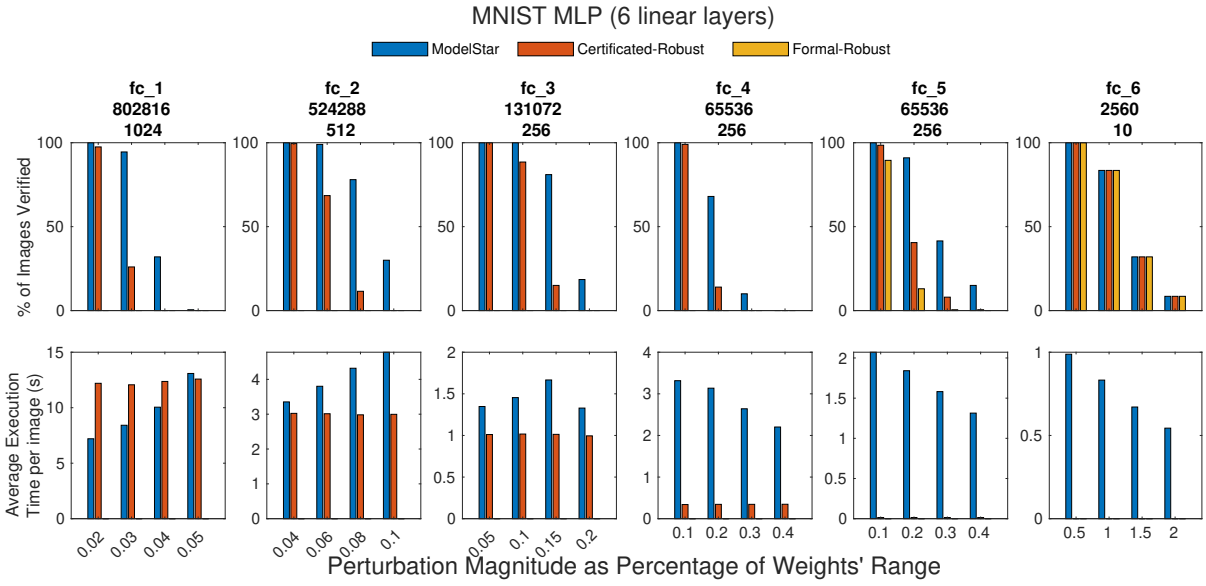

Fig. 6. Verification results of 6-layer MLP on 200 MNIST images. The top row of plots shows the percentage of images verified to be classified successfully despite perturbations, and the bottom row of plots shows average verification time per image. The number of weights and number of neurons in each fully-connected layer are displayed below the layer's name.

5.1.1 Single-Layer Perturbation. Figure 6 shows the verification results using ModelStar, Certificated-Robust (T.-W. Weng et al. 2020)[2] as well as Formal-Robust (Tsai et al. 2021a) when a single layer of the MLP is perturbed at a time. The percentage of images verified by our approach, ModelStar, is equal to or greater than that verified by the other approaches. For example, ModelStar has verified the safe classification of 189 out of the 200 images under test even when the entire weight matrix of the first hidden fully connected layer is perturbed by 0.03% of the layer's weights range, whereas Certificated-Robust (T.-W. Weng et al. 2020) was able to verify the correct classification of only 52 out of the 200 images (a difference of 68.5%). The cost of the improved robustness verification is paid in execution time: though the execution times of ModelStar appear to be better or similar to those of Certificated-Robust for the earlier layers, Certificated-Robust is faster than ModelStar for the latter layers. Overall, we conclude that ModelStar propagates a larger amount of information than Certificated-Robust, resulting in better verification but higher execution times.

The variation in execution times of ModelStar as perturbation magnitudes increase shows differing trends across different layers. This behavior is primarily influenced by the two components of the verification process: *reachability analysis* and *safety verification*. During reachability analysis, the reachable set of the perturbed layer is propagated through subsequent layers to obtain the output reachable set of the entire NN. This step dominates the execution time of the earlier layers because the reachable sets must traverse a greater number of non-linear (ReLU) layers. On the other hand, the *safety verification* phase involves solving LPs to determine if the NN's output reachable set intersects with the unsafe set. Since the unsafe set for classification consists of a union of half-planes with each half-plane representing an incorrect class, as soon as the reachable set is found to intersect

[2]The code for this approach was not available at the repository published in their paper, so we implemented it using the algorithm they provided.

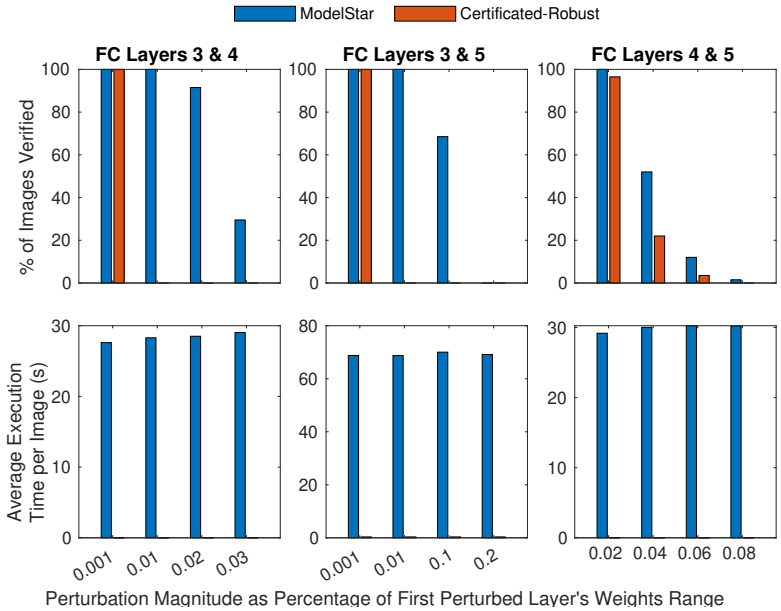

Fig. 7. Verification results of 6-layer MLP on 200 MNIST images, with two layers perturbed at a time. Top row of plots shows percentage of images verified and bottom row shows corresponding average execution time per image. In a number of perturbation scenarios, Certificated-Robust—due to its high conservativeness—failed to verify the MLP's classification robustness for any image, leading to only ModelStar having a non-zero "% of Images Verified" in those scenarios.

with any unsafe half-plane, the solution of all LPs is terminated. Since increase in perturbation magnitudes increases the number of unsafe half-planes intersecting with the NN's reachable set, safety verification is faster for higher perturbation magnitudes. As the reachable sets of latter layers pass through fewer non-linear layers, safety verification dominates the verification time for the latter layers.

The number of perturbed weights in a layer also affects the verifiable robustness. In this experiment, all weights have been perturbed for one layer at a time, and Figure 6 shows the number of weights in each fully-connected layer below the layer's name. The top plots in the figure show that the layers with fewer weights have been verified to be robust to higher magnitudes of full-layer perturbation as compared to the layers with more weights. The verifiable robustness also decreases due to the over-approximation introduced during propagation through the non-linear ReLU layers. Hence, fully-connected layer 4 has lower verifiable robustness than fully-connected layer 5 for the same perturbation magnitudes. Since there is no ReLU layer after the sixth fully connected layer (output layer), the verification results are similar for all approaches regarding perturbations in this layer.

5.1.2 Multi-Layer Perturbation. Figure 7 shows the verification results when two layers of the MNIST MLP are perturbed simultaneously. The ∞-norm bounds on the perturbation magnitudes are displayed in the figure as a **percentage of the weights range of the first of the two perturbed layers**, and the entire weight matrices of both perturbed layers are subject to perturbations within this ∞-norm bound. The 200 input images used for this experiment are the same as the single-layer case. We compare the multi-layer verification results of ModelStar with Certificated-Robust (T.-W. Weng et al. 2020) only as the authors of Formal-Robust (Tsai et al. 2021a) do not present a way to compute bounds on the output of the NN in the case of multi-layer perturbation. Once again, we observe that the percentage of images verified by our approach, ModelStar, is consistently equal to or

greater than that verified by Certificated-Robust, particularly when perturbations occur in earlier layers. For instance, when the third and fourth fully connected layers are subject to perturbations, for certain perturbation magnitudes, ModelStar verifies robust classification of all 200 images, whereas Certificated-Robust is unable to verify any of them. Hence, this experiment demonstrates that ModelStar's encoding of multi-layer perturbations using the results from **Theorem 4.3**—despite over-approximation—results in much higher verifiable robustness than the state-of-the-art bound. While Certificated-Robust is faster, the improved robustness bounds offered by ModelStar make it the preferable choice in most cases.

5.1.3 Detailed Comparison for a Single Image. Now we zoom in on the verification of a single image of the digit 0, from the second case in Figure 7, i.e., when the third and fourth fully-connected layers of the MLP are perturbed by 0.01% of the weights range of the third layer. Since perturbations result in uncertainty in the NN outputs, we determine the worst-case upper and lower bounds on the outputs of the NN using both approaches. These bounds are shown as ranges in Figure 8, computed using ModelStar as well as Certificated-Robust (T.-W. Weng et al. 2020). The figure shows that the bounds obtained using Certificated-Robust are more conservative than the ones obtained using ModelStar, allowing ModelStar to verify the robustness where Certificated-Robust failed to do so.

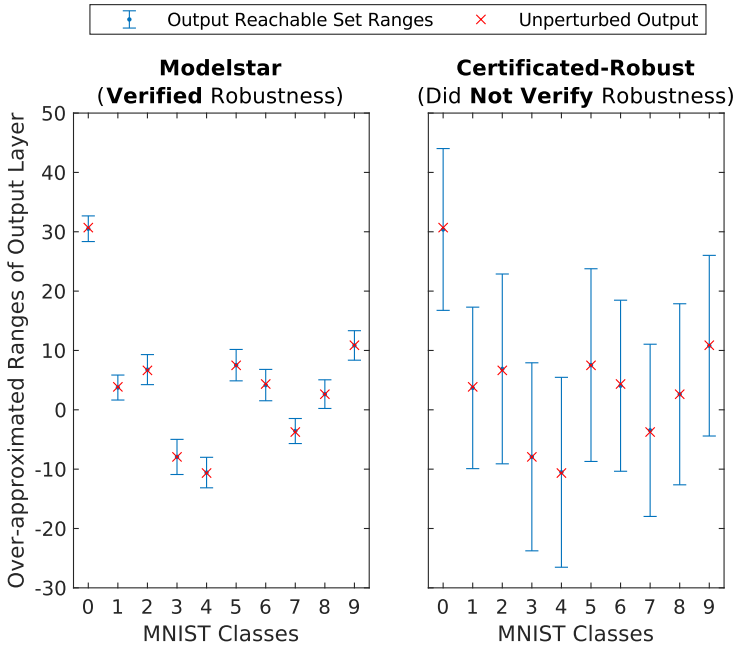

Fig. 8. Comparison of ModelStar and Certificated-Robust for verification of an MNIST MLP with two perturbed layers for a single image of digit 0. The figure shows the ranges of the MLP's outputs when the weight matrices of the third and fourth fully-connected layers of the MLP are subject to perturbations within 0.01% of the weights range of the third fully-connected layer.

5.2 Convolutional Neural Network (MNIST)

This section presents results for the verification of a CNN trained on the MNIST dataset. The CNN under consideration has 5 convolutional layers and 3 fully connected layers, with the complete architecture given by

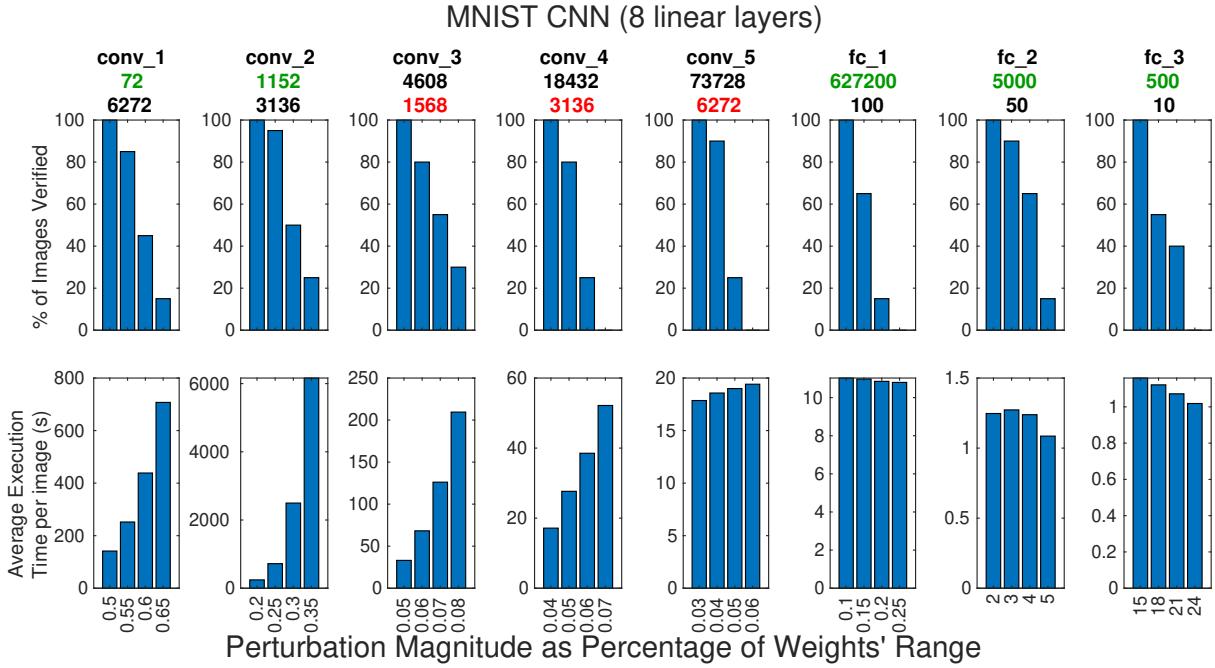

Fig. 9. Verification results of 8-layer CNN on 20 MNIST images. Top row of plots shows percentage of images verified and bottom row shows average execution time per image. At the top of the plots of each layer, the layer's name, number of weights and number of neurons are displayed.

Tran, Bak, et al. (2020b) under the name MNIST_Large. Weight perturbations in the convolutional layer result in highly dense generator tensors as compared to fully-connected layers and result in higher execution times. Hence, we run this experiment for 20 images from the MNIST dataset, i.e., 2 images per class. For this CNN, we focus on the verifiable robustness obtained using ModelStar due to a lack of alternative frameworks capable of formal verification against convolutional layer perturbations. We perturb all weights in one layer of the CNN at a time and measure robustness in terms of the percentage of images that are classified correctly despite perturbations. The resulting verifiable robustness, along with the execution time, is shown in Figure 9. We are interested in the maximum perturbation magnitude, relative to the weights range of each layer, to which the CNN is found to be completely robust. For the first and second convolutional layers, these perturbation magnitudes are 0.5% and 0.2%, respectively. However, these magnitudes lie around 0.05%, 0.04% and 0.03% for the third, fourth and fifth convolutional layers, respectively. This sudden increase in conservativeness (decrease in verifiable robustness) is primarily due to the use of Theorem 4.2 for reducing the number of generators in these layers' reachable sets from the number of weights to the number of neurons. The motivation for doing so is that the number of neurons in these layers is much smaller than the number of weights. Hence, Theorem 4.2 offers a trade-off between verification capability and resource consumption. To facilitate analysis, Figure 9 shows the number of weights and neurons of each layer. The use of Theorem 4.2 is indicated in the figure by the red colored display of the number of neurons of the corresponding layers. The layers whose initial reachable sets do not have any over-approximation have their weights displayed in green.

The execution time trends for the verification of the CNN, as shown in Figure 9, are similar to those of the MLP trends in Figure 6. However, the execution times for the first convolutional layer are smaller than the execution times for the second convolutional layer, primarily due to the drastic difference in the number of weights (72 vs. 1152). This suggests that propagating a star set with fewer generators through more non-linear layers can be less time-consuming compared to propagating a star set with a larger number of generators through fewer non-linear layers. We also observe an exponential increase in the execution time of the *reachability analysis* step, primarily due to the exponential growth of the non-linearity of the ReLU layer (Tran, Bak, et al. 2020b). This trend can be seen prominently in the execution times corresponding to the first, second and third convolutional layers in Figure 9 due to the relatively larger number of ReLU layers after these layers.

5.3 Fully-Connected Layers of VGG19

In this section, we evaluate the verification capability of ModelStar on VGG19, a CNN with 16 convolutional layers and 3 fully connected layers. VGG19 has been trained on the ImageNet-1k dataset which contains about 1.3 million images across 1000 classes (Simonyan and Zisserman 2015). Our experiments focus on assessing the robustness of VGG19 by perturbing its fully-connected layers individually, an entire layer at a time. We utilize 20 images from various ImageNet classes to conduct these tests. As in the case of the MNIST MLP, ModelStar exhibits superior verification capabilities compared to existing methods. For instance, when the first fully-connected layer is perturbed by 0.01% of the layer's weights range, ModelStar successfully verifies the correct classification of 15 out of 20 images, whereas Certificated-Robust verifies only 8. However, this tighter robustness bound comes at the cost of execution time: while ModelStar's execution times are approximately twice those of Certificated-Robust in some cases, they significantly increase with larger perturbations. Specifically, a 1% perturbation in the entirety of the first fully connected layer results in an average verification time of 120 minutes per image. Consequently, we limit our experiments to 20 images due to these extensive execution times. Additionally, the latter convolutional layers of VGG19 have a high neuron count, necessitating numerous LP solutions when propagating star sets. This leads to increased memory and execution time requirements for verification against perturbations in the earlier layers. Therefore, in our experiments, we limit the perturbations in VGG19 to the weights of its fully-connected layers.

5.4 Multi-Layer Perceptron (ACAS Xu)

We also evaluate ModelStar for an ACAS Xu NN (Katz et al. 2017). ACAS Xu is a set of 45 NNs used as lateral thrust advisories for airborne collision-avoidance systems. Their inputs include properties of ownship and intruder, such as their speeds, heading angles and the distance between them. All 45 ACAS Xu NNs are fully-connected ReLU networks with identical architecture: 6 hidden layers with 50 neurons each and one output layer with 5 outputs, each output advising a direction of horizontal thrust. From the set of 45 NNs, the selection of the appropriate NN for thrust direction advice at any given time is based on the previously used advisory and the time until loss of vertical separation.

To ensure safety of the flight advisory NN, 10 safety tests have been proposed, henceforth called *properties* (Katz et al. 2017). Each property defines NN input/output specifications that ensure desirable behavior such as collision avoidance. However, not all of these properties are relevant to all ACAS Xu NNs. We choose one MLP from the ACAS Xu set, subject it to weight perturbations and verify how robustly this MLP satisfies the 6 properties relevant to this NN. Figure 11 shows the results of perturbing one layer of the MLP at a time, with all weights in the layer subject to bounded perturbations. Though the safety properties specify ranges of inputs, we test the MLP's robustness against weight perturbations as it processes a nominal, unperturbed input. We obtain this nominal input by taking the average of the upper and lower bounds provided in each property's input specifications. Figure 11 compares the verification capabilities of ModelStar with Certificated-Robust (T.-W. Weng

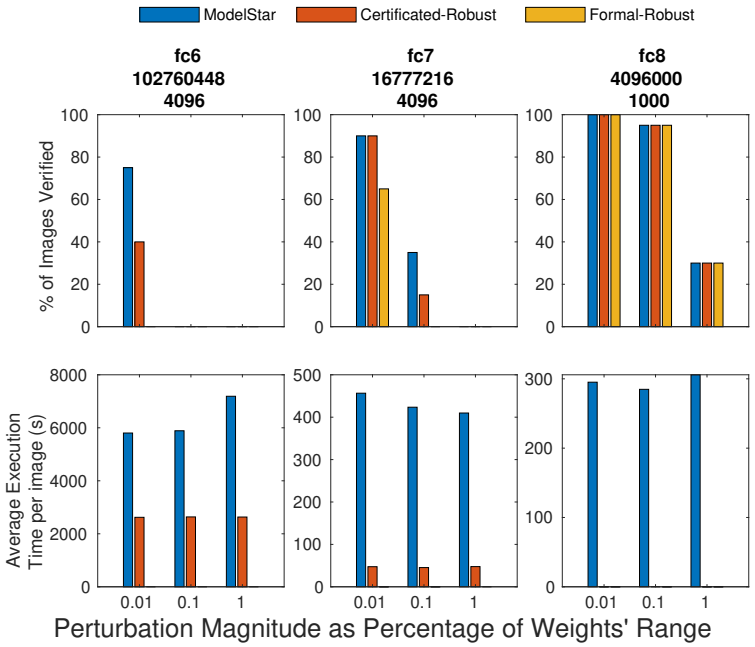

Fig. 10. Verification results of VGG19 on 20 ImageNet images. Top row of plots shows percentage of images verified and bottom row shows average execution time per image. At the top of the plots of each layer, the layer's name, number of weights and number of neurons are displayed.

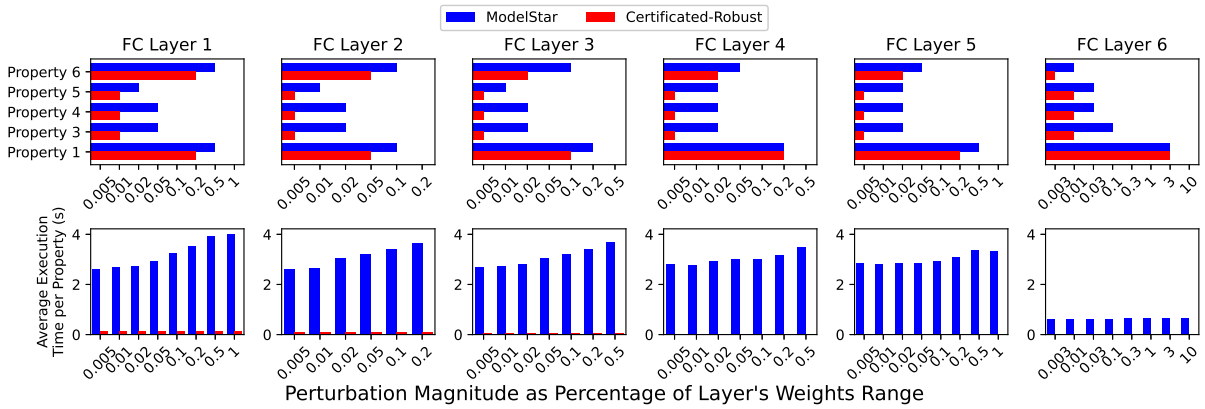

Fig. 11. Verification results of an ACAS Xu MLP (6 hidden layers) for 6 safety properties (Katz et al. 2017). The top plots show bars spanning the perturbation magnitude bounds that are verified to be "safe" in the context of the respective property. The bottom bar plots show the average execution time per property, for verification against each perturbation magnitude.

et al. 2020). It can be seen that ModelStar generally outperforms Certificated-Robust in verification capability, though at the cost of higher execution times.

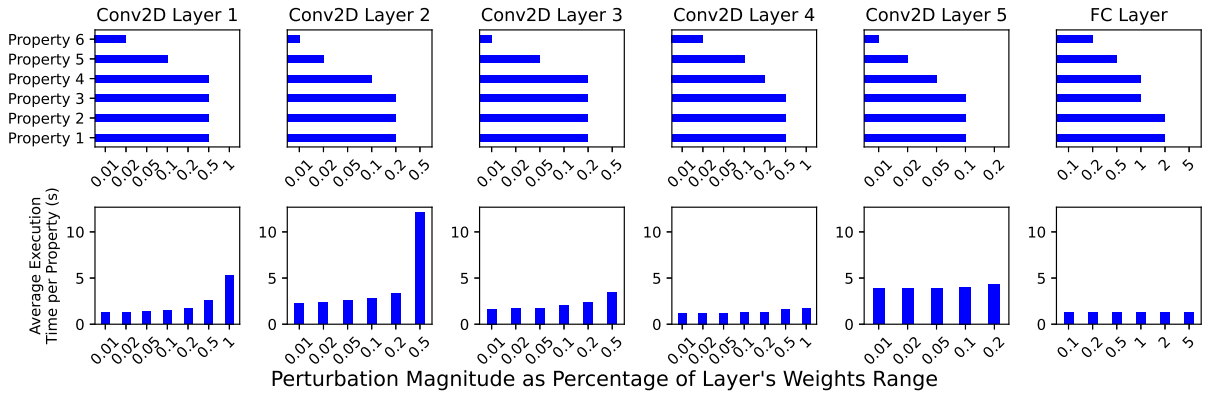

Fig. 12. Verification results of a Collins RUL CNN (6 layers) for 6 safety properties (Katz et al. 2017). The top plots show bars spanning the perturbation magnitude bounds that are verified to be "safe" in the context of the respective property. The bottom bar plots show the average execution time per property, for verification against each perturbation magnitude.

5.5 Collins Remaining Useful Life (RUL) CNN

Next, we evaluate ModelStar on a CNN used for prediction of remaining useful life (RUL) of aircraft components (Kirov and Rollini 2023). RUL prediction enables predictive maintenance instead of preventative maintenance, resulting in reduced costs. However, the safety-critical nature of the application makes it an interesting case study for the formal verification community. The CNN is trained on hourly timeseries data consisting of:

- Component condition indicators extracted from vibration sensors attached to bearings.
- Nature of the flight mission, including states of ascent or descent as well as the load on the component.
- Flight environment, such as desert or non-desert.

We choose the CNN trained on an input window size of 40 for our experiments. This CNN has 5 convolutional layers and one fully connected layer.

Kirov and Rollini (2023) suggest numerous safety properties for RUL CNNs that capture desirable RUL prediction behavior. We choose 6 such properties for our experiments:

- Properties 1-3 check monotonicity, i.e., whether an increase in a critical condition indicator results in an increase in the NN's output (or vice versa).
- Properties 4-6 check bounded-input bounded-output robustness, i.e., small, bounded perturbations in the input produce only small, bounded variations in the output.

As in the case of the ACAS Xu safety properties, we average the upper and lower bounds of the input specifications of each property to obtain a nominal input per property for use in our experiments because our study is focused on weight perturbation analysis instead of input perturbations. As in the case of the MNIST CNN, we have no existing benchmark to compare the verification results against. Rather, we simply choose properties that are satisfied without perturbations and then introduce increasing, layer-wide perturbations in a single-layer at a time. The verification and execution time results of the experiments are shown in Figure 12 against each perturbation magnitude. To indicate the extent of perturbation despite which the NN safely satisfies a property, the verification plots show bars spanning the verifiably safe ranges of perturbation magnitude. As expected, the properties become unsatisfied with increase in perturbation magnitude, due to the increasing size of the output

reachable set. The exponential execution time trends of the earlier layers are similar to those of the MNIST CNN as already discussed in Section 5.2.

So far, we have shown experimentally that ModelStar surpasses the state-of-the-art approaches in verification capabilities, but it has certain limitations, which we discuss in the next section.

5.6 Limitations and Discussion

While ModelStar advances robustness verification against model perturbations in NNs, certain limitations remain.

Firstly, the *exact* mode of propagating star sets through an NN provides precise robustness verification for given weight perturbations but is computationally expensive when applied to a large number of perturbations. To address this, the *approximate mode* of star set propagation, as demonstrated in our experiments, offers a more practical alternative. However, since it relies on over-approximation, the resulting robustness bounds may exhibit a noticeable gap when compared to the empirical performance of randomly perturbed NNs. Nevertheless, this gap is expected to be relatively smaller when compared to the actual worst-case perturbation bounds, and further research is required to quantify this difference more precisely.

Secondly, while ModelStar captures the impact of individual weight perturbations with greater granularity, it does not provide a closed-form expression for their effect on the network's outputs, as is the case in some other approaches (Tsai et al. 2021a; T.-W. Weng et al. 2020). The absence of a closed-form solution is a trade-off between computational feasibility and ModelStar's ability to explicitly model and verify perturbations at the level of individual weights, thereby enabling a more detailed robustness analysis.

Additionally, ModelStar supports the verification of robustness across multiple perturbed layers simultaneously. However, this results in products of predicate variables, as discussed in Section 4.3, leading to increased over-approximation and a corresponding reduction in the lower bound on verifiable robustness. A similar challenge arises when extending reachability analysis to multiple input samples. While multiple inputs can be encoded as a star set at the input layer, this approach introduces additional over-approximation effects due to interactions between predicate variables. Developing more sophisticated techniques to integrate weight perturbation analysis with input perturbation verification remains an important direction for future research. Finally, as observed in our experiments, ModelStar's execution time increases with the number of perturbed weights. This effect is particularly pronounced in convolutional layers where denser generator sets are formed and neuron counts are high, requiring the solution of a larger number of LPs to propagate the resulting reachable sets through the subsequent ReLU layers (Tran, Bak, et al. 2020b).

Extension: Additionally, there is an interesting way to extend this work for quantitative analysis of the robustness of an NN. ModelStar performs qualitative robustness verification through reachability analysis. For classification tasks, this involves determining a reachable set for given perturbations and confirming if the correctly classified output consistently ranks the highest. For instance, consider the scenario depicted in the first plot of Figure 8, where the output range for the correct class (0) does not overlap with the output range of any other class, ensuring correct classification despite weight perturbations. However, overlaps in output ranges can potentially render the robustness of the NN unverifiable due to over-approximation. Such a case is shown in Figure 13. Here, although the output ranges for labels 0 and 2 overlap, it is possible for the actual distribution of outputs within this overlapping area to be minimal, as shown by the example distribution. It indicates that most outputs lie within the non-overlapping portions, suggesting a high likelihood of correct classification despite the perturbations. While probability-based quantification of NN robustness has been developed for input perturbations (Tran, Choi, et al. 2023), this approach has not yet been extended to weight perturbation analysis. This quantitative approach provides a more realistic assessment of NN robustness and is particularly vital in scenarios involving over-approximate reachability analysis, where output sets tend to be conservative. We plan to integrate this

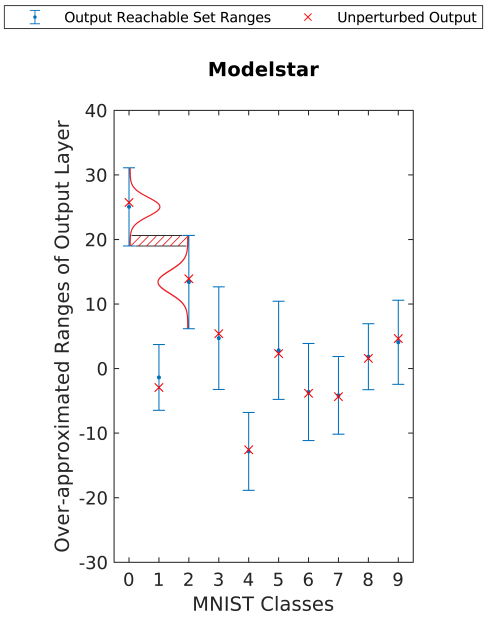

Fig. 13. The output values of the NN follow a probability distribution, and this figure illustrates the mass in the intersecting ranges of possible distributions for the correct class and an incorrect class.

quantified verification into our ModelStar framework, enhancing its capability to deliver more precise robustness assessments.

6 Conclusion

In this paper, we introduced ModelStar, a reachability-based framework for verifying the robustness of NNs against weight perturbations. While exact robustness verification remains computationally challenging for large perturbation spaces, our experiments demonstrated that ModelStar's over-approximate verification approach significantly improves upon existing methods by providing a tighter lower bound on NN robustness in fully connected layers. Additionally, ModelStar extends formal verification to perturbations in convolutional layers, a capability not supported by prior verification frameworks. Future work will focus on further enhancing ModelStar's scalability and precision. Specifically, we aim to refine the over-approximation of the star set to reduce conservativeness while preserving computational efficiency. Furthermore, we will explore methods to quantify NN robustness across the perturbation space, enabling a more comprehensive robustness assessment. Finally, we plan to investigate the impact of NN size reduction techniques on ModelStar's verification performance to improve its applicability to large-scale networks.

Acknowledgments

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 2220426, 2220401 and 2325416 and the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-23-C-0518. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA or NSF. A preliminary version of this work was presented at the IEEE Conference on Artificial Intelligence (2025) (Zubair,

T. T. Johnson, et al. 2025), but without the theoretical proofs and extensive experimental evaluation provided in this work.

References

J. Bai, B. Wu, Z. Li, and S.-T. Xia. 2023. "Versatile weight attack via flipping limited bits." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45, 11, 13653–13665.

S. Bak and H.-D. Tran. 2022. "Neural network compression of ACAS Xu early prototype is unsafe: Closed-loop verification through quantized state backreachability." In: *NASA Formal Methods Symposium*. Springer, 280–298.

F. Boudardara, A. Boussif, P.-J. Meyer, and M. Ghazel. 2022. "Interval weight-based abstraction for neural network verification." In: *International Conference on Computer Safety, Reliability, and Security*. Springer, 330–342.

C. Brix, S. Bak, T. T. Johnson, and H. Wu. Dec. 28, 2024. *The Fifth International Verification of Neural Networks Competition (VNN-COMP 2024): Summary and Results*. (Dec. 28, 2024). arXiv: 2412.19985 [cs]. http://arxiv.org/abs/2412.19985.

K. Cai, M. H. I. Chowdhuryy, Z. Zhang, and F. Yao. 2024. "Deepvenom: Persistent dnn backdoors exploiting transient weight perturbations in memories." In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2067–2085.

N. Cheney, M. Schrimpf, and G. Kreiman. 2017. *On the robustness of convolutional neural networks to internal architecture and weight perturbations*. arXiv: 1703.08245.

X. Echeberria-Barrio, A. Gil-Lerchundi, R. Orduna-Urrutia, and I. Mendialdua. 2022. "Optimized parameter search approach for weight modification attack targeting deep learning models." *Applied Sciences*, 12, 8, 3725.

K. Fang, Q. Tao, Y. Wu, T. Li, J. Cai, F. Cai, X. Huang, and J. Yang. 2024. "Towards robust neural networks via orthogonal diversity." *Pattern Recognition*, 149, 110281.

M. Fazlyab, T. Entesari, A. Roy, and R. Chellappa. 2024. "Certified robustness via dynamic margin maximization and improved lipschitz regularization." *Advances in Neural Information Processing Systems*, 36.

N. Fuengfusin and H. Tamukoh. 2024. "NaN Attacks: Bit-Flipping Deep Neural Network Parameters to NaN or Infinity." In: *2024 1st International Conference on Robotics, Engineering, Science, and Technology (RESTCON)*. IEEE, 33–37.

S. Ganapathy, J. Kalamatianos, K. Kasprak, and S. Raasch. 2017. "On characterizing near-threshold SRAM failures in FinFET technology." In: *Proceedings of the 54th Annual Design Automation Conference 2017*, 1–6.

D. Goldberg. 1991. "What every computer scientist should know about floating-point arithmetic." *ACM computing surveys (CSUR)*, 23, 1, 5–48.

C. Gongye and Y. Fei. 2024. "One Flip Away from Chaos: Unraveling Single Points of Failure in Quantized DNN s." In: *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 332–342.

D. Grimm, D. Tollner, D. Kraus, Á. Török, E. Sax, and Z. Szalay. 2024. "A numerical verification method for multi-class feed-forward neural networks." *Expert Systems with Applications*, 247, 123345.

A. Izycheva and E. Darulova. 2017. "On sound relative error bounds for floating-point arithmetic." In: *2017 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 15–22.

G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. 2017. "Reluplex: An efficient SMT solver for verifying deep neural networks." In: *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer, 97–117.

D. Kirov and S. F. Rollini. 2023. "Benchmark: remaining useful life predictor for aircraft equipment." In: *International Conference on Bridging the Gap between AI and Reality*. Springer, 299–304.

E. Korkmaz. 2021. "Investigating vulnerabilities of deep neural policies." In: *Uncertainty in Artificial Intelligence*. PMLR, 1661–1670.

S. Kundu, S. Banerjee, A. Raha, S. Natarajan, and K. Basu. 2021. "Toward functional safety of systolic array-based deep learning hardware accelerators." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29, 3, 485–498.

S. Levin and J. C. Wong. Mar. 2018. *Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian*. Accessed: May 11, 2024. The Guardian. (Mar. 2018). https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe.

L. Li, T. Xie, and B. Li. 2023. "Sok: Certified robustness for deep neural networks." In: *2023 IEEE symposium on security and privacy (SP)*. IEEE, 1289–1310.

S. Li, X. Wang, M. Xue, H. Zhu, Z. Zhang, Y. Gao, W. Wu, and X. S. Shen. 2024. "Yes, One-Bit-Flip Matters! Universal DNN Model Inference Depletion with Runtime Code Fault Injection." In: *Proceedings of the 33th USENIX Security Symposium*.

Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. 2021. "A survey of convolutional neural networks: analysis, applications, and prospects." *IEEE transactions on neural networks and learning systems*, 33, 6999–7019, 12.

J. Liu, Y. Xing, X. Shi, F. Song, Z. Xu, and Z. Ming. 2024. "Abstraction and refinement: Towards scalable and exact verification of neural networks." *ACM Transactions on Software Engineering and Methodology*, 33, 5, 1–35.

D. M. Lopez, S. W. Choi, H.-D. Tran, and T. T. Johnson. 2023. "NNV 2.0: the neural network verification tool." In: *International Conference on Computer Aided Verification*. Springer, 397–412.

S. Maji, K. Lee, C. Gongye, Y. Fei, and A. P. Chandrakasan. 2024. "An energy-efficient neural network accelerator with improved resilience against fault attacks." *IEEE Journal of Solid-State Circuits*, 59, 3106–3116, 9.

M. H. Meng, G. Bai, S. G. Teo, Z. Hou, Y. Xiao, Y. Lin, and J. S. Dong. 2022. "Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective." *IEEE Transactions on Dependable and Secure Computing*, Early Access. doi:10.1109/TDSC.2022.3179131.

A. Michel, S. K. Jha, and R. Ewetz. 2022. "A survey on the vulnerability of deep neural networks against adversarial attacks." *Progress in Artificial Intelligence*, 11, 2, 131–141.

O. Özdenizci and R. Legenstein. 2022. "Improving robustness against stealthy weight bit-flip attacks by output code matching." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13388–13397.

J. Perez-Cerrolaza et al.. 2024. "Artificial intelligence for safety-critical systems in industrial and transportation domains: A survey." *ACM Computing Surveys*, 56, 7, 1–40.

H. Pourmehrani, J. Bahrami, P. Nooralinejad, H. Pirsiavash, and N. Karimi. 2024. "FAT-RABBIT: Fault-Aware Training towards Robustness AgainstBit-flip Based Attacks in Deep Neural Networks." In: *2024 IEEE International Test Conference (ITC)*. IEEE, 106–110.

P. Prabhakar and Z. Rahimi Afzal. 2019. "Abstraction based output range analysis for neural networks." *Advances in Neural Information Processing Systems*, 32.

C. Qian, M. Zhang, Y. Nie, S. Lu, and H. Cao. 2023. "A Survey of bit-flip attacks on deep neural network and corresponding defense methods." *Electronics*, 12, 4, 853.

M. H. Rahman, S. Laskar, and G. Li. 2024. "Investigating the impact of transient hardware faults on deep learning neural network inference." *Software Testing, Verification and Reliability*, 34, 4, e1873.

P. Rech. 2024. "Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions." *IEEE Transactions on Nuclear Science*, 71, 4, 377–404.

F. F. d. Santos, N. Cavagnero, M. Ciccone, G. Averta, A. Kritikakou, O. Sentieys, P. Rech, and T. Tommasi. 2025. "Improving Deep Neural Network Reliability via Transient-Fault-Aware Design and Training." *IEEE Transactions on Emerging Topics in Computing*, 13, 3, 829–840. doi:10.1109/TETC.2024.3520672.

H. Shin, M. Kang, and L.-S. Kim. 2022. "Fault-Free: A Framework for Analysis and Mitigation of Stuck-at-Fault on Realistic ReRAM-Based DNN Accelerators." *IEEE Transactions on Computers*, 72, 7, 2011–2024.

K. Simonyan and A. Zisserman. 2015. "Very deep convolutional networks for large-scale image recognition." In: *3rd International Conference on Learning Representations (ICLR 2015)*. Computational and Biological Learning Society.

K. Smagulova, L. Bacha, M. E. Fouda, R. Kanj, and A. Eltawil. 2024. "Robustness and Transferability of Adversarial Attacks on Different Image Classification Neural Networks." *Electronics*, 13, 3, 592.

D. Stutz, N. Chandramoorthy, M. Hein, and B. Schiele. 2020. *On mitigating random and adversarial bit errors*. arXiv: 2006.13977.

X. Sun, Z. Zhang, X. Ren, R. Luo, and L. Li. 2021. "Exploring the vulnerability of deep neural networks: A study of parameter corruption." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35(13), 11648–11656.

H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson. 2020a. "Verification of deep convolutional neural networks using imagestars." In: *International conference on computer aided verification*. Springer, 18–42.

H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson. 2020b. *Verification of Deep Convolutional Neural Networks Using ImageStars*. arXiv: 2004.05511 (cs.LG).

H.-D. Tran, S. Choi, H. Okamoto, B. Hoxha, G. Fainekos, and D. Prokhorov. 2023. "Quantitative verification for neural networks using probstars." In: *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, 1–12.

H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson. 2019. "Star-based reachability analysis of deep neural networks." In: *Formal Methods–The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings 3*. Springer, 670–686.

Y.-L. Tsai, C.-Y. Hsu, C.-M. Yu, and P.-Y. Chen. 2021a. "Formalizing generalization and adversarial robustness of neural networks to weight perturbations." *Advances in Neural Information Processing Systems*, 34, 19692–19704.

Y.-L. Tsai, C.-Y. Hsu, C.-M. Yu, and P.-Y. Chen. 2021b. "Non-singular adversarial robustness of neural networks." In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3840–3844.

K. Wang, Z. Chen, X. Dang, X. Fan, X. Han, C.-M. Chen, W. Ding, S.-M. Yiu, and J. Weng. 2023. "Uncovering hidden vulnerabilities in convolutional neural networks through graph-based adversarial robustness evaluation." *Pattern Recognition*, 143, 109745.

L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. 2018. "Towards fast computation of certified robustness for relu networks." In: *International Conference on Machine Learning*. PMLR, 5276–5285.

T.-W. Weng, P. Zhao, S. Liu, P.-Y. Chen, X. Lin, and L. Daniel. 2020. "Towards certificated model robustness against weight perturbations." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34(04), 6356–6363.

H. Wu et al.. 2024. "Marabou 2.0: a versatile formal analyzer of neural networks." In: *International Conference on Computer Aided Verification*. Springer, 249–264.

L. Xiang, X. Zeng, Y. Niu, and Y. Liu. 2019. "Study of sensitivity to weight perturbation for convolution neural network." *IEEE Access*, 7, 93898–93908.

T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal, S. Agarwal, and M. J. Marinella. 2022. "On the accuracy of analog neural network inference accelerators." *IEEE Circuits and Systems Magazine*, 22, 4, 26–48.
P. Xu, F. Wang, W. Ruan, C. Zhang, and X. Huang. 2023. "Sora: Scalable black-box reachability analyser on neural networks." In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.
D. Yadron and D. Tynan. June 2016. *Tesla driver dies in first fatal crash while using autopilot mode*. Accessed: May 11, 2024. The Guardian. (June 2016). https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk.
Z. Yan, X. S. Hu, and Y. Shi. 2022. "Computing-In-Memory Neural Network Accelerators for Safety-Critical Systems: Can Small Device Variations Be Disastrous?" In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 1–9.
Y. Yao, T. Han, J. Yu, and M. Xie. 2024. "Uncertainty-aware deep learning for reliable health monitoring in safety-critical energy systems." *Energy*, 291, 130419.
L. Yu, Y. Wang, and X.-S. Gao. 2023. "Adversarial parameter attack on deep neural networks." In: *International Conference on Machine Learning*. PMLR, 40354–40372.
H. Zhao, H. Hijazi, H. Jones, J. Moore, M. Tanneau, and P. Van Hentenryck. 2024. "Bound tightening using rolling-horizon decomposition for neural network verification." In: *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 289–303.
M. U. Zubair, T. Johnson, K. Basu, and W. Abbas. Feb. 2026. *ModelStar: Neural Network Verification Framework for Weight Perturbation Analysis*. (Feb. 2026). doi:10.5281/zenodo.18701712.
M. U. Zubair, T. T. Johnson, K. Basu, and W. Abbas. 2025. "Verification of Neural Network Robustness Against Weight Perturbations Using Star Sets." In: *2025 IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 637–642.
M.-M. Zühlke and D. Kudenko. 2024. "Adversarial robustness of neural networks from the perspective of Lipschitz calculus: A survey." *ACM Computing Surveys*, 57, 6, 1–41.

A Proof of Theorem 4.3

Proof. The perturbed weight tensors W_i' and b_i' can be expressed as:

$$W_i' = W_i + \sum_{k=1}^{q} \beta_k M_{Wk},\ b_i' = b_i + \sum_{k=1}^{q} \beta_k M_{bk}$$

where β_k is a scalar bounded variable, and M_{Wk} and M_{bk} are the corresponding perturbation maps.

Let the input to the linear layer under consideration be the star set $\mathcal{I}$ having the form defined in (2). The resulting unperturbed output of the layer is $L_i(\mathcal{I}) = W_i \odot \mathcal{I} + b_i$. Thus, the perturbed output is

$$\begin{aligned} L_i'(\mathcal{I}) &= W_i' \odot \mathcal{I} + b_i' \\ L_i'(\mathcal{I}) &= \left(W_i + \sum_{k=1}^{q} \beta_k M_{Wk}\right) \odot \left(c + \sum_{j=1}^{m} \alpha_j v_j\right) + \\ &\quad b_i + \sum_{k=1}^{q} \beta_k M_{bk} \end{aligned}$$

We denote the product $\alpha_j \beta_k$ by a new variable γ_{jk}. Hence, $L_i'(\mathcal{I})$ can be expressed as:

$$\begin{aligned} L_i'(\mathcal{I}) = L_i(c) + \sum_{j=1}^{m} \alpha_j W_i \odot v_j + \\ \sum_{k=1}^{q} \beta_k \left(M_{Wk} \odot c + M_{bk}\right) + \sum_{k=1}^{q} \sum_{j=1}^{m} \gamma_{jk} M_{Wk} \odot v_j \end{aligned} \tag{8}$$

Let us first discuss the first three terms on the right hand side of (8), as these terms are linear: the first term, $L_i(c)$, is the nominal operation of the i-th layer which updates the center point c of the star set (as $L_i(c) = W_i \odot c + b_i$). The second term, $\sum_{j=1}^{m} \alpha_j W_i \odot v_j$, is processed by the existing implementation of ImageStar (Tran, Bak, et al.

2020a) by multiplying or convolving W_i with each v_j, in accordance with the definition of $\odot$. The third term, $\sum_{k=1}^{q} \beta_k \, (M_{Wk} \odot c + M_{bk})$, is dealt with in the same fashion as the single-layer perturbation case in Section 4.2.

Now we address the non-linearity in the fourth term, i.e.,

$$\sum_{k=1}^{q} \sum_{j=1}^{m} \gamma_{jk} M_{Wk} \odot v_j = \sum_{k=1}^{q} \sum_{j=1}^{m} \alpha_j \beta_k M_{Wk} \odot v_j$$

This term constitutes products of the predicate variables α_j and β_k. We deal with this by treating each product $\alpha_j \beta_k$ as a new predicate variable γ_{jk} whose bounds are computed from the bounds of the multiplying predicate variables. If the ranges of α_j and β_k are $[l_1, u_1]$ and $[l_2, u_2]$, respectively, the lower and upper bounds of γ_{jk} are the minimum and maximum, respectively, of the four products $l_1 l_2, l_1 u_2, u_1 l_2$ and $u_1 u_2$.

The products of all α_j with all β_k result in an explosion of the new predicate variables γ_{jk}. Thus, it is very memory-consuming if each γ_{jk} is represented as an individual variable in the output reachable set. Instead, we use Algorithm 1 to obtain a bounding box over-approximation of all generators corresponding to the product variables γ_{jk}. In doing so, we discard the constraints imposed on γ_{jk} due to its constrained factor α_j (α_j is constrained by $C\alpha \leq d$, as in (2)). Discarding the constraints on the variables γ_{jk} introduces over-approximation in the resulting reachable set. However, doing so is required in order to apply Algorithm 1 for the over-approximation of the generators corresponding to the γ_{jk} variables, since Algorithm 1 only applies to such generators whose variables are unconstrained except lower and upper bounds. Consequently, the mq generators in the fourth term of (8) are reduced to (at most) n generators $(\alpha'_r, e_r), r = 1, \ldots, n$.

Let us define:

$$\begin{aligned} c' &= L_i(c) \\ v'_j &= W_i \odot v_j \\ u_k &= M_{Wk} \odot c + M_{bk} \end{aligned}$$

Now, the reachable set $\mathcal{R}'_i$ of the i-th layer can be soundly over-approximated by the following set:

$$\begin{aligned} \mathcal{R}''_i = \{y \mid y = c' + \sum_{j=1}^{m} \alpha_j v'_j + \sum_{r=1}^{n} \alpha'_r e_r + \sum_{k=1}^{q} \beta_k u_k, \\ C\alpha \leq d, \ \underline{\alpha'} \leq \alpha' \leq \overline{\alpha'}, \ \underline{\beta} \leq \beta \leq \overline{\beta}\} \end{aligned}$$

where $\underline{\beta}$ and $\overline{\beta}$ are the bounds of the perturbation vector $\beta = [\beta_1, \beta_2, \ldots, \beta_q]$, and $\underline{\alpha'}$ and $\overline{\alpha'}$ are the bounds of the perturbation vector $\alpha' = [\alpha'_1, \alpha'_2, \ldots, \alpha'_n]$. $\underline{\alpha'}$ and $\overline{\alpha'}$ are obtained through Algorithm 1 as discussed above. □

Received 21 April 2025; accepted 21 January 2026