Verification of Neural Network Robustness Against Weight Perturbations Using Star Sets

Muhammad Usama Zubair*, Taylor T. Johnson†, Kanad Basu*, Waseem Abbas*
*University of Texas at Dallas, Richardson, TX, USA
†Vanderbilt University, Nashville, TN, USA
{muhammadusama.zubair, kanad.basu, waseem.abbas}@utdallas.edu, taylor.johnson@vanderbilt.edu

***Abstract*—Neural network reliability is susceptible to perturbations in both inputs and model weights. While formal verification of neural network robustness against input perturbations is a hot topic, robustness analysis against weight perturbations remains an emergent area of research. Neural network weights are subject to hardware vulnerabilities and environmental changes, posing significant risks to system reliability in safety-critical applications such as autonomous driving and medical diagnostics. To address this issue, we introduce 'ModelStar,' a novel framework that utilizes reachability analysis to evaluate the robustness of deep neural networks (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 results demonstrate that ModelStar surpasses existing methods in analysing the impact of perturbations, verifying DNN robustness on up to 60% more samples in image classification tasks. Besides providing tighter robustness bounds than existing methods, ModelStar allows formal robustness verification against weight perturbations in any linear layer, a capability not supported by prior work, to our knowledge. This advancement marks a significant step towards the reliable use of DNN accelerators in safety-critical applications.**

***Index Terms*—Formal verification, neural networks, weight perturbation, reachability analysis.**

I. Introduction

Neural networks have become prevalent as the state-of-the-art in machine learning tasks over the past decade and are increasingly being utilized across a wide spectrum of areas [1]. These include safety-critical systems [2], such as autonomous driving [3] and aircraft collision avoidance systems [4]. Fatal accidents involving autonomous vehicles have already occurred [5], [6], underscoring the importance of rigorously analyzing the robustness of neural networks. Extensive research has been conducted on exposing the vulnerabilities of neural networks [7]–[9], as well as developing defenses against them [10]–[12]. To facilitate the use of neural networks in safety-critical systems, the safety verification of neural networks i.e., determining whether a neural network produces desirable results despite perturbations, has garnered increased attention in recent years [13]–[15]. However, most of these works focus on robustness verification against input perturbations, despite findings that neural networks are also vulnerable to weight perturbations, i.e., variations in the model parameters of DNNs [16]–[19].

Recently, there has been an increased interest in providing guaranteed lower bounds on the robustness of a neural network against weight perturbations, such as bounds on pair-wise class margins [20]. Another work has extended linear bounds on the outputs of neural networks–that previously accounted for input perturbations [21]–to handle weight perturbations [22]. However, there is still room for improvement. Our work improves this bound by extending reachability analysis–a technique used for robustness verification against input perturbations [23] –to handle weight perturbations. Previously, reachability analysis has been employed to construct a *set* representing *all possible states resulting from input perturbations* and propagate this set through the neural network layers. We extend this technique to account for weight perturbations by augmenting the states in the set as it passes through each layer containing perturbed weights. This adaptation is crucial because it propagates a larger amount of information through the layers of a neural network as compared to previous approaches [20], [22]. This enables ModelStar to provide tighter robustness guarantees, thereby providing higher confidence in the reliability of neural networks in practical applications. We utilize the ImageStar reachability analysis framework [23] due to its extensive support for propagating reachable sets through a variety of neural network layers. We refer to our extension, which integrates **model** perturbations into **star** reachable sets, as **ModelStar**.

The main contributions of our work include the following:

- We introduce **ModelStar**, a novel framework that incorporates **weight perturbations** of neural networks into reachable sets–specifically star sets–through the use of linearity as well as perturbation matrices, referred to as perturbation maps.
- Perturbation maps allow ModelStar to handle bounded perturbations affecting arbitrarily specific subsets of weights in **any linear layers**, for the purpose of analyzing the impact of these perturbations on the reliability of the neural network.
- We present complexity bounds on the size of the augmented reachable set, achieving exact representation of the outputs of a single perturbed layer, and an over-approximate representation of the outputs of multiple perturbed layers.
- Through rigorous numerical evaluation, we demonstrate that ModelStar is less conservative than state-of-the-art approaches [20], [22] in determining the safety of a neural network against weight perturbations. It verifies

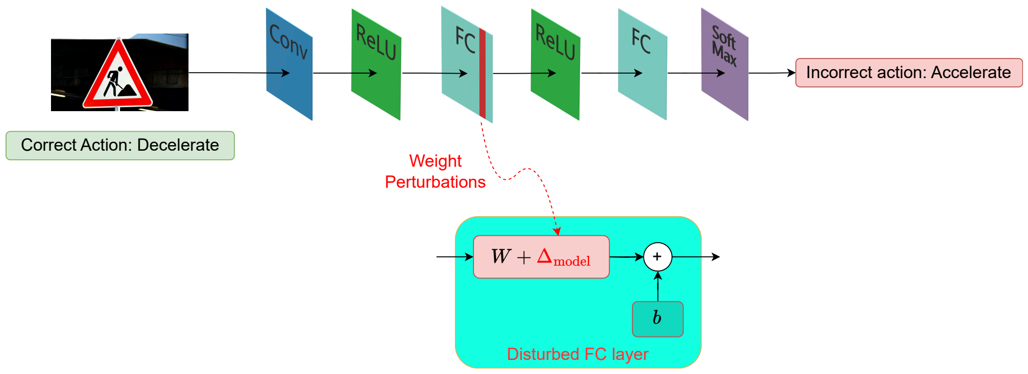
Conv
ReLU
FC
ReLU
FC
Soft Max
Incorrect action: Accelerate
Correct Action: Decelerate
Weight Perturbations
$W + \Delta_{\text{model}}$
+
b
Disturbed FC layer

Fig. 1. Model perturbations in neural networks 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.

the robustness of neural networks against perturbations in fully-connected layers for up to 60% more samples in image classification tasks.

The rest of this paper is organized as follows: Section II reviews related work on the significance of weight perturbations in neural networks and existing approaches to obtain robustness bounds against these perturbations. Section III outlines the problem setup. Section IV introduces the formulation of ModelStar. Section V presents numerical experiments comparing ModelStar with state-of-the-art approaches for verifying the robustness of fully connected neural networks against weight perturbations. Finally, Section VI concludes the paper.

II. Related Work

A. Impact of Weight Perturbations on NN Performance

Numerous studies in the literature have highlighted the sensitivity of neural networks to weight perturbations.

The authors of [24] investigate the sensitivity of the output of a convolutional neural network (CNN) to weight perturbations, across all possible inputs. The study in [25] employs 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. The authors of [16] concentrate on the perturbations caused by device variations in Computing-in-Memory (CiM) Deep Neural Network (DNN) accelerator architectures. They demonstrate that even minor device variations can lead to large performance drops in the worst-case scenario.

The work in [17] highlights the potential for modifying neural network operation for target samples with specific bit-flips, while preserving the performance of the neural network for the rest of the input space. The authors of [18] introduce another attack technique to decrease the classification accuracy for only a target cluster in the input data by altering the neural network weights. The authors of [19] prove the existence of such weight perturbations and present algorithms to determine them.

B. Robustness Bounds against Weight Perturbations

Recent research has sought to quantify the effects of norm-bounded perturbations in weights of neural networks.

The study in [22] focuses on the maximum possible norm bounded perturbation in the weights of a neural network such that the classification output remains correct i.e., the output for the correct class remains higher than the output for any other class. Since this is an NP-hard problem, they derive a lower bound on this perturbation, by utilizing linear upper and lower bound functions of the neural network function. They consider norm bounds on perturbations in one row of the weight matrix. The authors of [20] also consider norm-bounded weight perturbations and present a bound on the pair-wise class margin in their presence. They consider ∞-norm bounds on perturbations in entire matrices. To summarize, [22] presents a bound on the disturbance being applied, whereas [20] presents bounds on the neural network output as a result of the disturbances. We utilize the bounds presented in [22] and [20] for comparison with ModelStar in our numerical evaluation.

III. Preliminaries and Problem Setup

We define a feed-forward neural network $\mathcal{N}$ as a sequence of layers $\mathcal{N} = \{L_i\}$, $i = 1, 2, \ldots, n$, with activation layers indexed distinctly.

A. Reachable Sets

Our approach involves computing sets that encapsulate all possible outputs of a layer resulting from perturbations. These sets are then propagated through the remaining layers of the neural network, 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 1 (Reachable Set): Given a set $\mathcal{I}$ as an input to the network, the output reachable set $\mathcal{R}_i$ of the i-th layer is obtained by applying the layer's operation L_i to the output reachable set of the preceding layer [23].

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

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

A graphical illustration of the reachability analysis process is presented in Fig. 2. Next, we present the set structure used for reachability analysis in this paper, called the *star set*.

B. Star Sets

In order to expand the scope of our theorems, we present a broader definition of star sets than previous work [23]:

Definition 2 (Star Set): A star set $\mathcal{I}$ is a linear set defined by a central point c, and m basis tensors$v_j,\ j = 1, \dots, m$ 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 $\{v_1, v_2, \dots, v_m\}$, and P is a set of predicates, limited to linear constraints. The set is mathematically defined as follows:

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

where $x, c, v_j \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_k}$, $j = 1, \dots, m$, and α is a vector of m predicate variables $[\alpha_1, \alpha_2, \dots, \alpha_m]$. If $P(\alpha) \triangleq C\alpha \le d$ 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 Equation 1 comprises m generators.

The authors of [23] discuss the propagation of star sets through various neural network layers, including fully-connected, convolutional, ReLU, batch normalization, average pooling and max pooling layers.

C. Safety Verification

To evaluate the robustness of a neural network, we examine the intersection of its reachable set, $\mathcal{R}_n$, with a predefined set of unsafe outputs. If this intersection is empty, the neural network is considered robust. Conversely, a non-empty intersection indicates non-robustness. These scenarios are illustrated in Fig. 2. In verifying the robustness of a classification neural network, the correct class for an input is known. Therefore, for each incorrect class, we can define a half-plane using the inequality where the output corresponding to this incorrect class equals or exceeds the output for the correct class. The union of all such half-planes constitutes the unsafe set for the classification problem. In regression problems, the unsafe set includes all outputs that fall outside a specified radius around the correct output.

In this paper, we evaluate the performance of our robustness verification approach through an image classification task. To enable ModelStar to scale effectively to layers with a large number of parameters, we employ over-approximate reachable sets. Consequently, if the neural network's output reachable set, derived using ModelStar, intersects with the unsafe set, we declare the robustness of the neural network in that scenario as "unknown." This situation arises when ModelStar cannot conclusively determine the robustness of the neural network against specified perturbations.

IV. ModelStar: Reachability Analysis Framework for Model Perturbations

In the current section, after presenting our perturbation model, we discuss how weight perturbations in a linear layer of a neural network are translated to the generators in the star reachable set when there are no existing perturbations. Following that, we discuss the general case where perturbations already exist in the star set input to the perturbed linear layer.

A. Perturbed Linear Layers

In a neural network, perturbations in the i-th layer can change the output states 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 1). While non-linear layers can experience perturbations, our work focuses on perturbations in the parameters of *linear layers*:

Definition 3 (Linear Layer): Let the i-th layer of a neural network 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:

$$L_i(x) = W_i \odot x + b_i \tag{2}$$

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 Equation 2.

Definition 3 encompasses a variety of linear layers. When the layer under consideration is a fully-connected layer, $\odot$ represents matrix product. When the convolutional layer is concerned, $\odot$ represents matrix convolution. Our approach is applicable to other affine transformations, including those in image input and batch normalization layers [23]. However, these extensions are straightforward and not the focus of this paper.

The *weights range* of a linear layer is defined as the difference between the maximum and minimum scalar weights in the layer's weight tensor. The perturbations considered in our paper are scalar, bounded perturbations. For example, there may be m perturbations $\{\alpha_1, \alpha_2, \dots, \alpha_m\}$, each with its own lower and upper bound, affecting the parameters of a linear layer. A single perturbation may affect multiple parameters of a linear layer. To describe the affected parameters' locations, we associate with the perturbation a *perturbation map*:

Definition 4 (Perturbation Map): Let the i-th layer of a neural network be a linear layer with weight and bias tensors W_i and b_i. Consider a single perturbation α_j affecting W_i and b_i. The weight perturbation map of α_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 α_j, and zero otherwise. Similarly, the bias perturbation map of α_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 α_j, and zero otherwise.

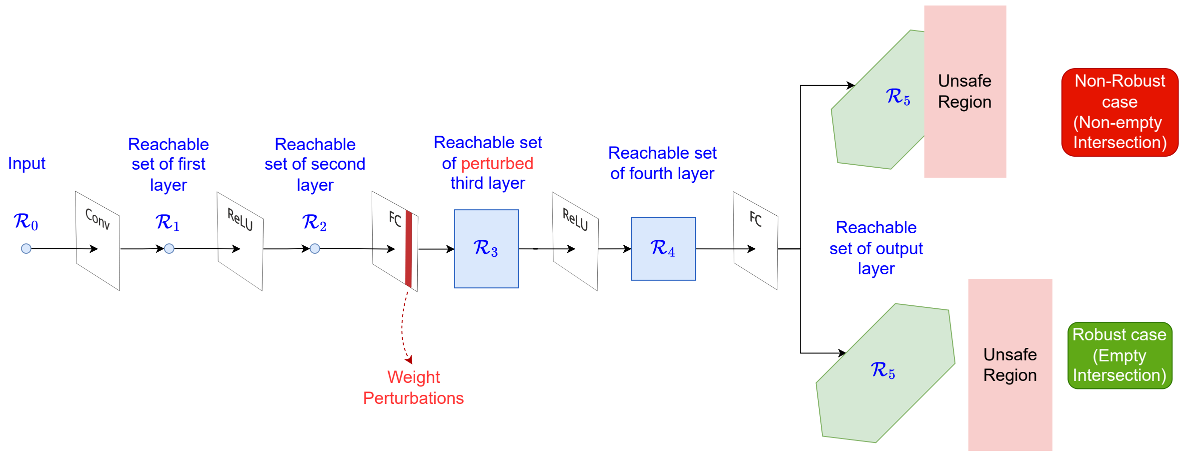
Input
$\mathcal{R}_0$
Conv
Reachable set of first layer
$\mathcal{R}_1$
ReLU
Reachable set of second layer
$\mathcal{R}_2$
FC
Weight Perturbations
Reachable set of perturbed third layer
$\mathcal{R}_3$
ReLU
Reachable set of fourth layer
$\mathcal{R}_4$
FC
Reachable set of output layer
$\mathcal{R}_5$
Unsafe Region
Non-Robust case (Non-empty Intersection)
$\mathcal{R}_5$
Unsafe Region
Robust case (Empty Intersection)

Fig. 2. Illustration of reachable set propagation through the neural network shown in Fig. 1. The terms 'Conv' and 'FC' stand for convolutional layer and fully-connected layer, respectively. Initially, the input to the neural network is a set containing a single element, and this remains true for the reachable sets up to the perturbed fully-connected layer. The perturbation expands the reachable set, which then propagates through the subsequent layers. The reachable set of the output layer is used for safety verification. If its intersection with the set of unsafe outputs is non-empty, the neural network is not robust to the given perturbations. If the intersection is empty, the neural network is robust. The figure shows both cases. Propagation through the softmax layer is unnecessary for safety verification against incorrect classification due to the monotonicity of the softmax layer.

Fig. 3 shows a 2×2 weight matrix subject to two perturbations α_1 and α_2. The perturbation map M_{W2} represents the effect of the perturbation α_2 on two weights simultaneously.

$$W' = \underbrace{\begin{array}{|c|c|}\hline 2 & 1 \\ \hline 3 & 5 \\ \hline\end{array}}_{W} + \alpha_1 \underbrace{\begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 1 \\ \hline\end{array}}_{M_{W1}} + \alpha_2 \underbrace{\begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0.5 \\ \hline\end{array}}_{M_{W2}}$$

Fig. 3. A 2×2 weight matrix W is subject to two bounded perturbations α_1 and α_2 resulting in the perturbed weight matrix $W' = W + \alpha_1 M_{W1} + \alpha_2 M_{W2}$. α_1 affects the weight at location $(2,2)$, and α_2 affects the weights at locations $(2,1)$ and $(2,2)$; the corresponding locations are shaded in red. M_{W1} and M_{W2} are example weight perturbation maps corresponding to perturbations α_1 and α_2, respectively.

B. Reachable Set for First Perturbed Layer

The following remark discusses the reachable set for the linear layer whose input is a set containing a single element. This scenario occurs when there are no perturbations in any prior layer and the input to the neural network is also a set containing a single element.

Remark 1: Let the i-th layer of a neural network be a linear layer (Definition 3) 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 the star set $\langle c, V, P \rangle$ comprising m generators (Definition 2). Here, $c = W_i \odot x + b_i$, V is the set of m basis tensors $\{v_1, v_2, \ldots, v_m\}$ with $v_j = M_{Wj} \odot x + M_{bj}$ (Definition 4), and $P(\alpha) = \alpha_l < \alpha < \alpha_u$ where the perturbation vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]$ is bounded by the vectors α_l and α_u.

Essentially, given a perturbation α_j and the locations of the weights and biases affected by α_j, Remark 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, Remark 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 neural network as demonstrated in Section V.

C. Reachable Set for the General Case

In general, the input to a linear layer is a star set (Definition 2). This scenario occurs if there are perturbations in a previous layer or in the input to the neural network. In this case, the predicate variables corresponding to the new perturbations in the current linear layer's weights multiply with the existing predicate variables in the input star set, leading to non-linear behavior. One way to deal with this is to employ a nonlinear structure, but this approach drastically increases complexity and does not allow for the use of elegant linear reachable set propagation techniques. The other approach, that we adopt in ModelStar, is over-approximation. This preserves the linearity of the reachable set at the cost of increased conservativeness. We present bounds on the size of the resulting reachable star set in the following remark:

Remark 2: Let the i-th layer of a neural network be a linear layer with unperturbed layer operation $L_i(x) = W_i \odot x + b_i$ (Definition 3). 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 (Definition 2) 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.

Owing to space limitations, we postpone the derivations of Remarks 1 and 2 to future work.

V. Experimental Evaluation and Discussion

In this section, we evaluate the verification capability of ModelStar over an MLP (Multi-Layer Perceptron) trained on the MNIST dataset[1]. We introduce varying amounts of perturbation in the weights of the fully-connected layers of the neural network. 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 finding the minimum tolerance threshold for programming weights to memristors in CiM devices using write-verify [16]. Unlike ModelStar, the weight-perturbation models in state-of-the-art approaches do not allow the analysis of the impact of perturbations on individual weights [20], [22]. Thus, for fair comparison with state-of-the-art, we perturb all weights in a layer of a neural network at a time, one perturbation affecting each weight. However, the ModelStar framework is capable of analysing perturbations based on any perturbation map that conforms to Definition 4. The experiment has been conducted in MATLAB running on a 64-core Ubuntu machine with 512GB RAM.

A. Verification of MNIST Multi-Layer Perceptron (MLP) Against Weight Perturbations

MNIST is a dataset containing 28×28 images of handwritten digits. It has 10 classes-one for each digit from 0 to 9. We consider a set of input images that are correctly classified by the neural network in the absence of perturbations. Thus, the safety of the neural network trained on MNIST is verified by counting the number of images safely classified by the neural network despite the presence of weight perturbations within certain thresholds.

The MLP 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. The MLP has been trained on the MNIST dataset with an accuracy of 99.72%. Each fully-connected hidden layer is followed by a ReLU layer. We attempt to verify the robustness of the neural network for 200 input images, 20 per class of the MNIST dataset.

Fig. 4 shows the verification results 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 other approaches, Certificated-Robust [22] and Formal-Robust [20]. For example, ModelStar has determined that 189 out of the 200 images under test would be correctly classified by the MLP even when the entire weight matrix of the first hidden fully connected layer is perturbed by 0.03%, whereas Certificated-Robust [22] 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 much faster than ModelStar for the latter layers.

[1]The code is available here: https://drive.google.com/drive/folders/1GtnlItJoZHLuQDolvWi9s1VnnKWGHTB-?usp=sharing

B. Limitations and Discussion

ModelStar vastly outperforms existing approaches [20], [22] due to its ability to account for individual weight perturbations, as well as the use of the tightest known convex over-approximation of the ReLU layer [23]. Despite this improvement in the state-of-the-art, ModelStar has some limitations. Firstly, the robustness of the neural network, as verified by ModelStar, is a lower bound, and ModelStar does not provide a closed form expression of the effects of the perturbations on the outputs of the neural network, whereas other approaches [20], [22] provide closed form results. The lack of a closed form is a price to be paid for verification with increased granularity i.e., ModelStar's ability to take individual weights' perturbations into account. Secondly, like existing approaches [20], [22], reachability analysis is currently limited to verification of the perturbed neural network for a single input at a time. Thirdly, as Fig. 4 indicates, ModelStar's execution time increases with increase in the number of perturbations. However, research in the reduction of the size and non-linearity of neural networks [26]–[28] will improve the scalability of ModelStar.

VI. Conclusion

In this paper, we have introduced ModelStar, a reachability-based method for verifying the robustness of neural networks against weight perturbations. To this end, ModelStar employs a novel approach through the use of perturbation maps and the linearity of parameterized layers. While exact robustness verification remains challenging for large perturbation spaces, our experiment demonstrates that ModelStar's over-approximate verification approach significantly outperforms existing methods by providing a tighter bound. However, the improved bound is still limited to a binary answer per input. Future work will quantify neural network robustness over the perturbation space to provide a more comprehensive measure of robustness. We also plan to explore the impact of weight perturbations in convolutional layers.

References

[1] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[2] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," *Information and Software Technology*, vol. 123, p. 106296, 2020.

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

[4] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–10.

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

[6] D. Yadron and D. Tynan. (2016, Jun) Tesla driver dies in first fatal crash while using autopilot mode. The Guardian. Accessed: May 11, 2024. [Online]. Available: https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk

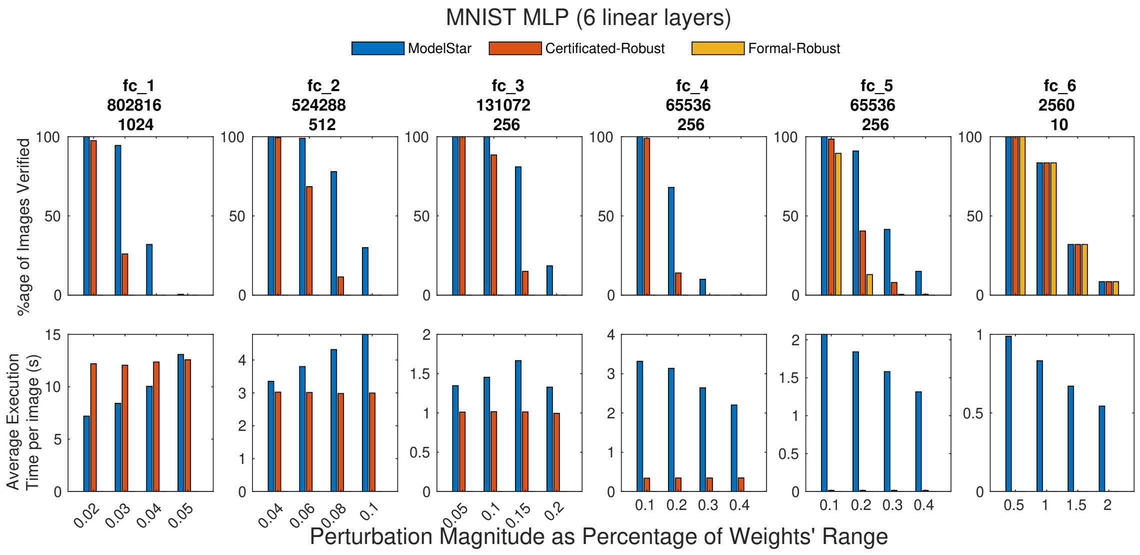

Fig. 4. Verification results of a 6-layer MLP on 200 MNIST images, with the entire weight matrix of a single layer perturbed at a time. The top row of plots shows the percentage of images verified to be classified successfully despite perturbations, and the bottom row of plots shows average verification time per image. Both are plotted versus the ∞-norm bound on the magnitude of perturbation. This perturbation bound is shown on the x-axis as a percentage of the range of weights of the corresponding layer. The number of weights and number of neurons in each fully-connected layer are displayed below the layer's name. The verification results are shown for three approaches: ModelStar, Certificated-Robust [22] and Formal-Robust [20].

[7] R. Sanchez-Matilla, C. Y. Li, A. S. Shamsabadi, R. Mazzon, and A. Cavallaro, "Exploiting vulnerabilities of deep neural networks for privacy protection," *IEEE Transactions on Multimedia*, vol. 22, no. 7, pp. 1862–1873, 2020.

[8] C.-J. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, and D. Lopez-Paz, "First-order adversarial vulnerability of neural networks and input dimension," in *International conference on machine learning*. PMLR, 2019, pp. 5809–5817.

[9] Z. Gu, W. Hu, C. Zhang, H. Lu, L. Yin, and L. Wang, "Gradient shielding: towards understanding vulnerability of deep neural networks," *IEEE transactions on network science and engineering*, vol. 8, no. 2, pp. 921–932, 2020.

[10] X. Zhang and M. Zitnik, "Gnnguard: Defending graph neural networks against adversarial attacks," *Advances in neural information processing systems*, vol. 33, pp. 9263–9275, 2020.

[11] X. Wang, S. Wang, P.-Y. Chen, Y. Wang, B. Kulis, X. Lin, and P. Chin, "Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses," *arXiv preprint arXiv:1908.07116*, 2019.

[12] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 402–433, 2020.

[13] M. H. Meng, G. Bai, S. G. Teo, Z. Hou, Y. Xiao, Y. Lin, and J. S. Dong, "Adversarial robustness of deep neural networks: A survey from a formal verification perspective," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[14] M. Kwiatkowska, "Safety and robustness for deep learning with provable guarantees," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020, pp. 1–3.

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

[16] Z. Yan, X. S. Hu, and Y. Shi, "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*, 2022, pp. 1–9.

[17] O. Özdenizci and R. Legenstein, "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*, 2022, pp. 13 388–13 397.

[18] X. Echeberria-Barrio, A. Gil-Lerchundi, R. Orduna-Urrutia, and I. Mendialdua, "Optimized parameter search approach for weight modification attack targeting deep learning models," *Applied Sciences*, vol. 12, no. 8, p. 3725, 2022.

[19] L. Yu, Y. Wang, and X.-S. Gao, "Adversarial parameter attack on deep neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 40 354–40 372.

[20] Y.-L. Tsai, C.-Y. Hsu, C.-M. Yu, and P.-Y. Chen, "Formalizing generalization and adversarial robustness of neural networks to weight perturbations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 692–19 704, 2021.

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

[22] T.-W. Weng, P. Zhao, S. Liu, P.-Y. Chen, X. Lin, and L. Daniel, "Towards certificated model robustness against weight perturbations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6356–6363.

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

[24] L. Xiang, X. Zeng, Y. Niu, and Y. Liu, "Study of sensitivity to weight perturbation for convolution neural network," *IEEE Access*, vol. 7, pp. 93 898–93 908, 2019.

[25] X. Sun, Z. Zhang, X. Ren, R. Luo, and L. Li, "Exploring the vulnerability of deep neural networks: A study of parameter corruption," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 648–11 656.

[26] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 913–14 922.

[27] F. Boudardara, A. Boussif, P.-J. Meyer, and M. Ghazel, "Innabstract: An inn-based abstraction method for large-scale neural network verification," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[28] D. Xu, N. J. Mozumder, H. Duong, and M. B. Dwyer, "Training for verification: Increasing neuron stability to scale dnn verification," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2024, pp. 24–44.