REAL-TIME CERTIFICATION OF ROBUSTNESS FOR *AE-DNNs* WITH THE DIFF MILP MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks (DNNs) are often brittle to small perturbations, which has led to extensive research to certify their robustness. Most existing methods focus on *local* robustness, i.e., in the neighborhood of fixed specific inputs. However, these techniques are often impractical for guaranteeing robustness to *real-time* inputs on embedded systems, due to excessive latency and computational cost.

To tackle real-time certification, we proceed in two steps. First, offline, we solve *global* robustness problems, which are significantly more complex than local robustness. Namely, we compute *bounds* on the difference of output values across different DNN decision classes under both L_∞- *and* L_1-perturbations. We then use the results online to perform real-time certification. On vanilla DNNs, however, global bounds are deemed overly pessimistic, because of inputs that are not In Distribution (ID). We propose alternative "AE-DNNs" to vanilla DNNs to use in exploitation, projecting inputs on an AutoEncoder (AE) latent space learnt from ID samples. With a sufficiently large latent space, ID inputs are only changed slightly and (Linear) AE-DNNs accuracy is close to Vanilla DNNs, with much smaller (3-15x) global bounds than Vanilla DNNs. To compute these bounds accurately, we develop novel *Diff* MILP models, explicitly representing the small *diff*erential variables between the input and its perturbation, instead of implicitly representing them as differences between 2 larger values (standard MILP encoding). Online, our method certifies in real-time over 70% of incoming images for AE-DNNs used *in exploitation*, across standard benchmarks (MNIST, Fashion MNIST and CIFAR-10), with L_1-perturbations of $1, 4, 2$, respectively, requiring only 0.5 ms of latency on a single CPU core.

1 INTRODUCTION

While deep neural networks (DNNs) have demonstrated remarkable capabilities, achieving human-like or even superior performance across a wide range of tasks, their robustness is often compromised by their susceptibility to input perturbations (Szegedy et al., 2014). This vulnerability has catalyzed the formal verification community to develop various methodologies, each presenting a unique trade-off between completeness and computational efficiency (Katz et al., 2019; 2017; Singh et al., 2019). This surge in innovation has also led to competitions, such as VNNComp (Brix et al., 2023), to systematically evaluate the performance of neural network verification tools. Notable examples include NNenum (Bak, 2021), Marabou (Katz et al., 2019; Wu et al., 2024), and PyRAT (Durand et al., 2022), as well as frameworks such as MnBAB (Ferrari et al., 2022) (which builds upon ERAN (Singh et al., 2019) and PRIMA (Muller et al., 2022)) and α, β-CROWN (Wang et al., 2021; Xu et al., 2021), based on branch-and-bound methodology (Bunel et al., 2020).

Most of these verification methods address *local* robustness: given a DNN, a specific input image, and a small neighborhood around this image, they verify whether all images in the neighborhood yield the same classification. This neighborhood is provided by a maximal perturbation of the input image, often an L_∞-norm constraint. Under this metric, every subpixel of the input image can vary in a very small range, typically $\frac{2}{255}$ (that is, 2 levels of gray/blue/red/green). Although it is not necessarily the most meaningful perturbation, L_∞ is the usual choice because it is perfectly linear.

Crucially, local robustness verification methods are far too computationally intensive for real-time decision-making pipelines. Consider an autonomous vehicle processing a live video feed: images

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of the feed cannot be certified as robust in a few ms on embedded hardware. This prevents the development of safety filters that could skip non-robust images and only consider certified images.

In this paper, we achieve *real-time* certification by leveraging *global* robustness bounds that hold across the entire input domain rather than restricting analysis to the neighborhood of a fixed input. Our framework follows a two-step procedure:

Global bound analysis. The first step is performed offline and computes global bounds on the maximum possible shift between output values of different decision classes due to a perturbation, following an approach similar to VHAGaR (Kabaha & Cohen, 2024). Specifically, for any two decision classes C and D and perturbation ε, we compute an upper bound $\bar{\beta}^{\varepsilon}_{C,D}$ on $\max_{I,I',|I-I'|\leq\epsilon}(x_C - x'_C + x'_D - x_D)$, where $x_{C,D}, x'_{C,D}$ are the output values of class C, D for images I and I', respectively. Unlike local robustness methods, which require repeating computationally expensive calls for each input, the global upper bound $\bar{\beta}^{\varepsilon}_{C,D}$ is computed offline once per network and remains valid across the entire input space.

Real-Time Verification. The second step occurs online and in real time, after the DNN's inference phase. For a given input image I, we identify the class C with the highest output value x_C, and check whether for every other class $D \neq C$, $x_C - x_D > \bar{\beta}^{\varepsilon}_{C,D}$. Thereby, we can formally certify that the image I is robust to perturbation ε. This check requires only a few dozen CPU instructions, which can be completed under 1 ms on a single embedded CPU core. When a frame fails this verification, one could either skip it or revert to a safer, degraded mode until a trustworthy, robust frame is received.

Our main contributions address the challenge of computing *global bounds* $\bar{\beta}^{\varepsilon}_{C,D}$ for classes (C, D):

1. *The AE-DNN architecture for tight bounds.* For vanilla DNNs, global bounds are particularly pessimistic (see Section 4), because inputs far away from the training dataset need to be accounted for. This implies that *no* techniques can ever produce manageable upper bound $\bar{\beta}^{\varepsilon}_{C,D}$ for vanilla DNNs. Indeed, on inputs not *In Distribution (ID)*, DNNs typically behave erratically (Karunanayake et al., 2025). We propose to use the AE-DNN architecture (Fig. 2 from Section 4) *in exploitation*, as an alternative to the original vanilla DNNs. The AE-DNN first uses an auto-encoder (AE) learnt from ID samples, projecting the large input space to a small latent space which aligns well with ID samples. It is then projected back to the full space before being input to the original DNN. Non-ID samples will be projected by the AE-DNN to the ID-compatible latent space, so that they do not create artificially large global bounds. We select the size of the latent space to ensure that the accuracies (which by definition only consider ID inputs) of the AE-DNN and DNN are within 1 %. In exploitation, the AE-DNN has similar characteristics than the original DNN, but global bounds are much better, enabling real-time certification for AE-DNNs. We experimented with Linear AEs (LAE): While not as powerful as non-linear AE, analysing them incur no complexity cost, and they do not require any change of architecture. Experimentally, the latent space is $>$ 30x smaller than the full space, leading to bounds $\bar{\beta}^{\varepsilon}_{C,D}$ 3-15x smaller than for vanilla DNNs.
2. *Efficient global bound computation.* We develop the *Diff* MILP encoding for the global robustness problem. In this model, the variables are the values of the perturbed neurons, as well as the differences between the original and the perturbed neuron values (called the *diff variables*, introduced in Paulsen et al. (2020)). We study and encode how the *diff variables* evolve after passing through a ReLU (Prop. 4); see Section 5.1. By comparison, the *classical* MILP model (Tjeng et al., 2019) employed in Kabaha & Cohen (2024); Wang et al. (2022a; 2023) computes the *diff variables* as the difference between 2 larger values. Both encodings are *asymptotically* exact (for infinite runtime), but the *Diff* model is more accurate in practice (iso finite runtime) due to better linear relaxation. Further, from the *Diff MILP model*, which is asymptotically exact, we develop two abstract MILP models, which are more efficient but also asymptotically less accurate than *Diff*, namely the *2b+1* and the *1b* models.

In terms of perturbations, we consider conjunctions of L_∞- and L_1-norms, which allow us to accurately describe perturbations. For instance, "each subpixel is perturbed by at most $\frac{50}{255}$ (L_∞) and the sum of the absolute values of perturbations over all subpixels is at most 1" (L_1-perturbation).

Experimentally, compared with the optimized versions of the classical MILP encoding (VHAGaR (Kabaha & Cohen, 2024) and ITNE (Wang et al., 2023)), the *Diff MILP model* computes upper bounds $\bar{\beta}^{\varepsilon}_{C,D}$ with smaller gaps to the lower bound (see Table 1). The abstractions *2b+1* and *1b*

variants offer different trade-offs, reaching better bounds than the *Diff MILP* model when the instance is very complex, or when runtime is limited (Table 3). Overall, using *Diff*, *2b+1*, and *1b* MILP models enables the real-time certification of $\geq 70\%$ of fresh images on LAE-DNNs used in exploitation, for an L_1-perturbation of $1, 4, 2$ over the MNIST, Fashion-MNIST, and CIFAR-10 datasets, respectively; see Table 4. The online process only adds ≈ 0.5 ms of latency per image, and ≈ 2000 images/second can be processed per CPU core.

This is a Concept and Feasibility paper. Our techniques are tested on several DNNs and datasets, but the DNN size is limited (though concrete examples are tackled). Linear AE and Linear and ReLU activation functions are the ones we experimented with. We strongly believe that the methodology is novel, impactful, and not intrinsically limited to these restrictions, which will eventually be lifted.

1.1 Related Work on Global Robustness

Some works focus on training DNNs to be more globally robust through dedicated architectures, (Leino et al., 2021; Zhang et al., 2022; Sun et al., 2022; Singla & Feizi, 2022), which results in drop in accuracy. Our LAE-DNN approach does not require any change of architecture and thus accuracy change is limited (see Section 4). Further, Leino et al. (2021); Chen et al. (2021); Singla & Feizi (2022) do not consider L_∞ or L_1 norm. Last, (Fu et al., 2024) considers restricted input space.

Some works (Bastani et al., 2016; Ruan et al., 2019; Gopinath et al., 2018) estimate global robustness bounds, but without hard verification certificates, e.g., using probabilistic guarantees (Levy et al., 2023; Mangal et al., 2019). Others (Katz et al., 2019; Wang et al., 2022a; 2023; 2022b) consider certifying restricted Lipschitz bound, to understand how the output can change given an input perturbation. However, they do not consider whether the classification changes. Closest to our work is VHAGaR (Kabaha & Cohen, 2024), which computes bounds $\bar{\alpha}^{\varepsilon}_{C,D}$ similar to our bounds $\bar{\beta}^{\varepsilon}_{C,D}$. Crucially, VHAGaR bounds are not small enough for real-time robustness, whereas our bounds are.

Technically, while Paulsen et al. (2020) does not tackle global robustness, it is one of the first papers to consider *differential* verification, introducing the *diff* variables that are used here and partly in Wang et al. (2023). However, Paulsen et al. (2020) did not consider MILP encoding. The MILP-based works (Kabaha & Cohen, 2024; Wang et al., 2022a; 2023) use the classical MILP encoding from Tjeng et al. (2019). ITNE (Wang et al., 2023) adds explicitly one *linear* constraint (Eq. (3) in Wang et al. (2023)) on the *diff variables* $y, \hat{y}$, implied by each of our MILP models *Diff*, *2b+1* and *1b* models. VHAGaR (Kabaha & Cohen, 2024) instead adds per neuron constraints (depending on the perturbation) between the (binary) variables to simplify the MILP problem, which is well-adapted to occlusion and patches but less to L_1-perturbations. None of these works consider L_1-perturbations.

2 Notations and Preliminaries

To simplify the notation, we restrict the presentation to feed-forward, fully connected ReLU Deep Neural Networks (DNN for short), where the activation function is $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}$ with $\mathrm{ReLU}(x) = x$ for $x \geq 0$ and $\mathrm{ReLU}(x) = 0$ for $x \leq 0$. An ℓ-layer DNN is provided by ℓ weight matrices $\boldsymbol{W}^i \in \mathbb{R}^{d_i \times d_{i-1}}$ and ℓ bias vectors $\boldsymbol{b}^i \in \mathbb{R}^{d_i}$, for $i = 1, \ldots, \ell$. We call d_i the number of neurons in the hidden layer $i \in \{1, \ldots, \ell - 1\}$, d_0 the input dimension, and d_ℓ the output dimension. Given an input vector $\boldsymbol{z}^0 \in \mathbb{R}^{d_0}$, denoting $\hat{\boldsymbol{z}}^0 = \boldsymbol{z}^0$, we define inductively the value vectors $\boldsymbol{z}^i, \hat{\boldsymbol{z}}^i$ at layer $1 \leq i \leq \ell$ with:

$$\boldsymbol{z}^i = \boldsymbol{W}^i \cdot \hat{\boldsymbol{z}}^{i-1} + \boldsymbol{b}^i \qquad \hat{\boldsymbol{z}}^i = \mathrm{ReLU}(\boldsymbol{z}^i).$$

The vector $\hat{\boldsymbol{z}}$ is called post-activation values, $\boldsymbol{z}$ pre-activation values, and $\boldsymbol{z}^i_j$ is used to denote the j-th neuron in the i-th layer. For $\boldsymbol{x} = \boldsymbol{z}^0$ the (vector of) input, we denote by $f(\boldsymbol{x}) = \boldsymbol{z}^\ell$ the output.

Mixed Integer Linear Programming (MILP) can encode ReLU DNNs faithfully (extensions to other activation functions, e.g., maxpool in Huang et al. (2020)): For an *unstable* neuron n, that is, with values $x \in [\mathrm{LB}(n), \mathrm{UB}(n)]$ with constants $\mathrm{LB}(n) < 0 < \mathrm{UB}(n)$, the value $\hat{x}$ of $\mathrm{ReLU}(x)$ can be encoded exactly in an MILP formula Tjeng et al. (2019) with one binary/integer variable a valued in $\{0, 1\}$, using constraints:

$$\hat{x} \geq x \wedge \hat{x} \geq 0 \wedge \hat{x} \leq \mathrm{UB}(n)a \wedge \hat{x} \leq x - \mathrm{LB}(n)(1 - a) \tag{1}$$

Proposition 1. *Tjeng et al. (2019) A solution $x, \hat{x}, a$ of the above MILP program (1) satisfies $\hat{x} = \mathrm{ReLU}(x)$, and $a = 1$ if $x > 0$ and $a = 0$ if $x < 0$ (both $a = 0, 1$ are possible if $x = 0$).*

LP relaxation: MILP instances can be linearly relaxed into an LP over-abstraction, where variables a originally restricted to integers in $\{0, 1\}$ (binary) are relaxed to real numbers in the interval $[0, 1]$, while maintaining the same encoding. Since solving LP instances takes polynomial time, this optimization is significantly more efficient. However, this efficiency comes at the cost of precision, often resulting in less stringent bounds Huang et al. (2020). This approach is termed the *LP relaxation*.

Classical MILP encoding for global robustness: In this paper, we consider the *global* verification problem, where we optimize over all images I and perturbations I' of I with $|I - I'| \leq \varepsilon$. Consider:

- $x_j, \hat{x}_j$, for neurons j with input image I; $x'_j, \hat{x}'_j$, for neurons j with input the perturbed I',
- $y_j = x_j - x'_j$ and $\hat{y}_j = \hat{x}_j - \hat{x}'_j$ the *diff variable* proposed by Paulsen et al. (2020).

The classical encoding Tjeng et al. (2019) uses linear variables $x_j, \hat{x}_j, x'_j, \hat{x}'_j$, as well as integer variables a_j, resp. a'_j for the encoding of $\hat{x}_j = \mathrm{ReLU}(x_j)$, resp. $\hat{x}'_j = \mathrm{ReLU}(x'_j)$, for all neuron j. Here are the **set of constraints** $\mathcal{M}^{classical}$ to encode global robustness with L_∞-perturbations $\leq \varepsilon$:

- for all *input* neuron j (in layer 0), $-\varepsilon \leq x_j - x'_j \leq \varepsilon$ [L_∞-perturbation $\leq \varepsilon$],
- for all neuron j from layer $i \geq 1$, $x_j = [\boldsymbol{W}^i \cdot \hat{\boldsymbol{z}}^{i-1}]_j + \boldsymbol{b}^i_j$ and $x'_j = [\boldsymbol{W}^i \cdot \hat{\boldsymbol{z}}'^{i-1}]_j + \boldsymbol{b}^i_j$,
- for all neuron j, $\hat{x}_j$, resp. $\hat{x}'_j$, use Eq. (1) with integer variable a_j, resp. a'_j.

Handling L_1: To encode an L_1-perturbation, we use the following encoding *without* extra binary variables. We use k (= number of input neurons) additional *linear* variables $(A_j)_{j \leq k}$, with constraints:

1. for all *input* neuron j (in layer 0), $A_j \geq x_j - x'_j$ and $A_j \geq x'_j - x_j$,
2. $\sum_{j \leq k} A_j \leq \varepsilon$ [L_1-perturbation $\leq \varepsilon$].

Proposition 2. *MOSEK ApS (2025) Constraints 1 and 2 above are equivalent to* $\sum_{j \leq k} |x_j - x'_j| \leq \varepsilon$.

The complexity of solving an MILP problem is bounded by $2^{bin} \times LP$ Huang et al. (2020), where bin is the number of binary variables and LP the P-time complexity of solving LP. No extra binary variables are used in the L_1 encoding. The MILP objective function is defined in the next section.

3 Real-time verification

As explained in the introduction, our objective is to compute upper and lower bounds for $\beta^{\varepsilon}_{C,D} = \max_{I,I',|I-I'|\leq\epsilon}(x_C - x'_C + x'_D - x_D) = \max_{|I-I'|\leq\varepsilon}(y_C - y_D)$, where x_A, resp. x'_A are the output values of class $A \in \{C, D\}$ from image I, resp. from the perturbed image I'. The associated MILP **objective function** is: Maximize $x_C - x'_C + x'_D - x_D$ subject to the constraints above.

Let $\bar{\beta}^{\varepsilon}_{C,D}$ be an (upper) bound and$\underline{\beta}^{\varepsilon}_{C,D}$ be a feasible solution of $x_C - x'_C + x'_D - x_D$, the pair $(\underline{\beta}^{\varepsilon}_{C,D}, \bar{\beta}^{\varepsilon}_{C,D})$ being obtainable by calling Gurobi once. We have $\underline{\beta}^{\varepsilon}_{C,D} \leq \beta^{\varepsilon}_{C,D} \leq \bar{\beta}^{\varepsilon}_{C,D}$.

Assume we have computed (offline) $\bar{\beta}^{\varepsilon}_{C,D}$ for each C, D. Now in real-time:

0 Run inference on the DNN to obtain the values x_D of output neuron for each class D. Let C be the class with maximal x_C, that is, the class predicted by the DNN. These are the basic inference operations standardly performed for classification by a DNN.

1 Compute for each class $D \neq C$ a bit $b_D = 1$ iff $x_D + \bar{\beta}^{\varepsilon}_{C,D} \geq x_C$.

2 Return "certified robust" iff $\bigvee_{D \neq C}(b_D) = 0$.

Proposition 3. *If "certified robust" is returned over input I, then for all perturbed input I' with $|I - I'| \leq \varepsilon$, the DNN has the same decision on I and on I', i.e., the DNN is robust around I.*

The proof or Prop. 3 can be found in Appendix B. The overhead (steps 1,2 above) on top of the standard inference step 0, is $3(k-1)$ operations, i.e,. linear in k, the number of classes. For $k = 10$ classes (cases of MNIST, Fashion MNIST and CIFAR-10), it amounts to 27 operations in total: $(k-1)$ ADD(+), $(k-1)$ COMP(<) and $k-1$ OR. This can be performed extremely fast, even on limited hardware.

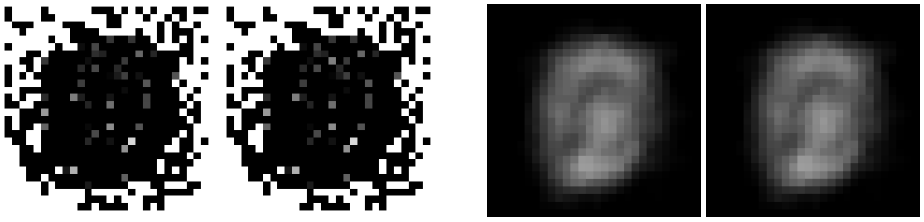

Figure 1: Left: pair (I,I') image and its perturbation (in a unique pixel at $x = 14$, $y = 8$ from top) maximizing $\underline{\beta}^{.5}_{6,8} = .518$ for DNN. Right: a pair (I,I') maximizing $\underline{\beta}^{.5}_{6,8} = .084$ for LAE-DNN.

4 THE AE-DNN ALTERNATIVE FOR REAL-TIME CERTIFICATION

Global upper bounds $\bar{\beta}$ obtained across the entire input space tend to be overly pessimistic for vanilla DNNs, since they must accommodate all possible inputs, including inputs *not* In (training) Distribution (ID) Karunanayake et al. (2025). Such samples display large value for $x_C - x'_C + x'_D - x_D$, see Fig. 1 (left) for MNIST. So no matter what method is employed to compute them, the upper bounds $\bar{\beta}^{\varepsilon}_{C,D}$ will be overly large, as they have to be at least as large as the values obtained for these samples.

We propose an efficient alternative to the original vanilla DNN: Construct AE-DNNs as follows and use them in exploitation instead. AE-DNNs exhibit very similar behavior on ID-inputs, which is critical for exploitation, but can differ vastly on *non* ID-inputs to avoid large bounds $\bar{\beta}^{\varepsilon}_{C,D}$. Hence, AE-DNNs achieve accuracy similar to vanilla DNNs while being more amenable to real-time certification. To do so, we train an Auto-Encoder (AE) on the ID dataset, obtaining an ID-compatible latent space. We then define the *AE-DNN architecture* (Fig. 2): first, AE-encode reduces to the latent space, then AE-decode produces the *projected input* (in full space), and last, call the DNN on this projected input.

In the following, we consider *Linear* AE-DNN (*LAE-DNN*), as the MILP model for *non-linear* AE-DNN needs additional (expensive) integer variables wrt (LEA-)DNN. Further, the original DNN architecture can be *retained* by absorbing the LEA layers of LEA-DNN into the DNN first layer's weights (Appendix C). Call dim the full-space dimension, and n the size of the latent space of the LAE trained on an ID dataset. LAE-decode is a linear function. By optimality, it can be assumed to be of rank n: it defines n linearly independent vectors of dimension $1 \times dim$. An optimal choice of LAE-encode is the projection of the input onto these n vectors. That is, calling LAE-decode and LAE-encode filters out features not in the latent space. Hence, on ID inputs, LAE-DNN behaves largely as the original DNN, but global bounds on the LAE-DNNs are smaller than on the original DNN, because of the projection to the latent space. Indeed, largest $x_C - x'_C + x'_D - x_D$ found on LAE-DNN is 6 times smaller than on DNN, and corresponds to more ID images; see Fig. 1 (right).

Training: We train the LAE on an ID dataset over various latent space dimensions n. We determine n by checking the LAE-DNN accuracy (i.e., the $\%$ of images whose predicted class matches the ground truth). We set n as small as possible (for efficiency) while ensuring that the LAE-DNN accuracy loss is minimal ($\leq 1\%$) wrt the original DNN. Such an n exists as LAE-decode(LAE-encode)=Id for $n = dim$, and thus LAE-DNN=DNN. For MNIST, FMNIST, CIFAR10, $n \leq \frac{dim}{30}$ (see Section 6.2).

Offline: Global bounds $\bar{\beta}$ are computed on LAE-DNN (on the full input space). The upper bounds computed are better (by a factor 3 to 15) for LAE-DNN than for DNN (Table 3 in Section 6.2).

Online Exploitation: LAE-DNN performs inference, and the global bounds of LAE-DNN are used, certifying in real-time robustness for ≈ 4 times larger ε than for DNN (Table 4 in Section 6.3). The prediction from LAE-DNN is used, as this is the one we have robustness guarantees about.

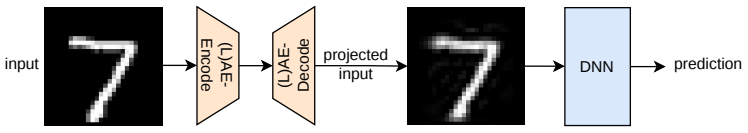

Figure 2: (L)AE-DNN architecture, filtering out non-ID features before applying DNN.

5 DIFF MILP MODELS FOR GLOBAL ROBUSTNESS

5.1 THE EXACT *Diff* MILP ENCODING FOR GLOBAL ROBUSTNESS

We now present our main methodological contribution, the *Diff* MILP encoding. It considers variables $x'_j, \hat{x}'_j, y_j, \hat{y}_j$, where $y_j, \hat{y}_j$ are the *Diff variables* (Paulsen et al., 2020), with $y_j = x_j - x'_j$ and $\hat{y}_j = \mathrm{ReLU}(x_j) - \mathrm{ReLU}(x'_j)$. Note that, since the image I and the deformation I' have perfectly symmetrical roles and assuming that γ_j is an *upper* bound of y_j, then $-\gamma_j$ is a *lower* bound for y_j.

To encode the equation describing $\hat{x}'_j$, we use the MILP constraints from the classical encoding (1), including one binary variable a'_j, similar to ITNE Wang et al. (2023). However, unlike ITNE, the *Diff* MILP model encodes the equation for $\hat{y}_j$ directly, without resorting to the classical encoding from $x_j (= y_j + x'_j)$ to $\mathrm{ReLU}(x_j)$ and using $\hat{y}_j = \mathrm{ReLU}(x_j) - \mathrm{ReLU}(x'_j)$, which is correct but does not encode explicitly that $\hat{y}_j$ is small when y_j is small, since both ReLUs can be large and they are encoded independently, see also Paulsen et al. (2020). Instead, we study the function from (x'_j, y_j) to $\hat{y}_j$, and encode it as the new *Diff* model.

We first display on Figure 3 the function $f(x'_j, y_j) = \hat{y}_j$ for $y_j \in [-\gamma_j, \gamma_j]$. When $x'_j \leq 0$ and $x_j = x'_j + y_j \leq 0$, we have $\hat{y}_j = \mathrm{ReLU}(x_j) - \mathrm{ReLU}(x'_j) = 0 - 0 = 0$. For $x'_j \geq 0$ and $x_j = x'_j + y_j \geq 0$, we have $\hat{y}_j = \mathrm{ReLU}(x_j) - \mathrm{ReLU}(x'_j) = x_j - x'_j = y_j$. Between these two linear planes, we have two more planes presenting intermediate cases, so 4 linear planes in total.

Recall first that a'_j is the binary variable used in the classical MILP encoding for $\hat{x}'_j = \mathrm{ReLU}(x'_j)$. Its value $a' = a'_j$ is settled by Prop. 1, reused in the MILP encoding of $\hat{y}_j = \mathrm{ReLU}(x_j) - \mathrm{ReLU}(x'_j)$:

Proposition 4. *Assuming $y_j \in [-\gamma_j, \gamma_j]$ and $x'_j, y_j + x'_j \in [\alpha_j, \beta_j]$ with $\alpha_j < 0 < \beta_j$, we have that $\hat{y}_j = \mathrm{ReLU}(x_j = y_j + x'_j) - \mathrm{ReLU}(x'_j)$ is the solution of the MILP program:*

$$
\begin{array}{ll}
y_j + x'_j \leq a\beta_j & y_j + x'_j \geq (1-a)\alpha_j \\
x'_j \leq a'\beta_j & x'_j \geq (1-a')\alpha_j \\
\hat{y}_j \leq a\gamma_j & \hat{y}_j \geq -a'\gamma_j \\
\hat{y}_j \leq y_j + x'_j - (1-a)\alpha_j & \hat{y}_j \geq y_j + x'_j - a'\beta_j \\
\hat{y}_j \leq -x'_j + a\beta_j & \hat{y}_j \geq -x'_j + (1-a')\alpha_j \\
\hat{y}_j \leq y_j + (1-a)\gamma_j & \hat{y}_j \geq y_j - (1-a')\gamma_j
\end{array}
$$

where $a, a' \in \{0,1\}$ are binary variables, a' being shared with the classical MILP encoding for $\hat{x}'_j$.

To encode both $\hat{y}_j$ and $\hat{x}'_j$, the *Diff* encoding uses 2 binary variables, similar to the classical encoding.

Intuitively, the *diff values* will be small as they will be carrying the difference between the image I and its slight perturbation I'. The LP relaxation of $\mathcal{M}^{classical}$ will miss this totally, while the LP relaxation of the Diff model contains such important bounds, including:

$$
\frac{y_j - \gamma_j}{2} \leq \hat{y}_j \leq \frac{y_j + \gamma_j}{2} \tag{2}
$$

This is a simplification (due to symmetry) of the additional linear constraint (3) of Wang et al. (2023).

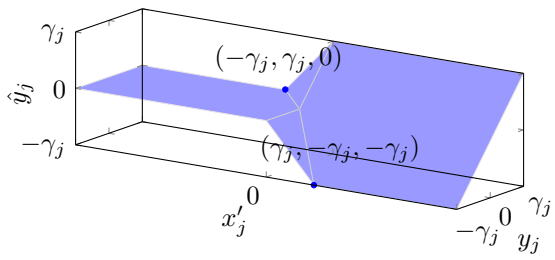

Figure 3: Function $(y_j, x'_j) \mapsto \hat{y}_j = \mathrm{ReLU}(x_j = y_j + x'_j) - \mathrm{ReLU}(x'_j)$

5.2 ABSTRACTIONS OF *Diff*: THE *2b+1* AND *1b* MODELS

We propose abstractions of the *Diff* encoding: the binary variables a' appearing in Prop. 4 can be decoupled from the binary variable a'_j appearing in the classical encoding of $\hat{x}'_j = \mathrm{ReLU}(x'_j)$. This would mean having 3 binary variables, which would be too costly. The variable a'_j used for x'_j is thus linearly relaxed, while both variables a and a' used in the encoding of $\hat{y}_j$ are kept binary. We call this the *2b+1* model. The decoupling between a' and a'_j removes constraints in the MILP encoding, which helps the solver. However, the *2b+1* model is not as accurate as the *Diff* model asymptotically.

To simplify further and limit the number of (binary) variables, we introduce the efficient *1b* model, which only uses the *diff* variables $y_j, \hat{y}_j$, and do not consider variables $x'_j, \hat{x}'_j$.

Proposition 5. *Given $y_j \in [-\gamma_j, \gamma_j]$, the range of values $\hat{y}_j$ can take (see Figure 4 in Appendix) corresponds to solutions of the following* 1b *MILP program with binary variable $a \in \{0, 1\}$:*

$$\hat{y}_j \leq a\gamma_j \qquad \hat{y}_j \geq y_j - a\gamma_j$$
$$\hat{y}_j \geq (a-1)\gamma_j \qquad \hat{y}_j \leq y_j + (1-a)\gamma_j$$

6 EXPERIMENTAL EVALUATION

We implemented our code in Python 3.8 and Gurobi 9.52 to solve the MILP problems. We conducted our evaluation on an AMD Threadripper 7970X (32 cores@4.0GHz) with 256 GB of main memory. ITNE Wang et al. (2023) was reimplemented using the same path as our code: the MILP variable x_j uses the classical MILP encoding plus the extra linear constraint Eq. (2). VHAGaR's code from `https://github.com/ananmkabaha/VHAGaR` was run on the same hardware.

We consider 4 pretrained Vanilla DNNs on 3 different datasets (MNIST, Fashion-MNIST, and CIFAR10). Namely, MNIST-FC, a fully-connected DNN from `https://github.com/eth-sri/eran` ("5x100"), and 3 convolutional DNNs: MNIST-Conv, FMNIST-Conv, and CIFAR10-Conv, from Kabaha & Cohen (2024) ("conv1"). We will also consider 3 LAE-DNNs (see Section 6.2), and a regression task (Appendix E).

6.1 COMPARISON FOR GLOBAL ROBUSTNESS WITH L_∞ BETWEEN *Diff*, VHAGAR AND ITNE

To be able to compare with VHAGaR, we use their benchmarks: VHAGaR bounds $\alpha_{C,D}$ computing the perturbation necessary to switch an image classified as class C into an image classified as class D (see Kabaha & Cohen (2024) and appendix B) instead of β from section 3, perturbation $L_\infty = 0.05$, VHAGaR convolutional DNNs, for which the VHAGaR reduction rules of binary variables are

$L_\infty = 0.05$	Timeout	MILP model:	VHAGaR	ITNE	*Diff*	*2b+1*	*Best*
MNIST-Conv LB $\underline{\alpha}$= 9.51	1000s	up. Bound $\overline{\alpha}\downarrow$	20.80	12.32	*11.14*	**10.95**	***10.85***
		(% of gap $\downarrow$)	(120%)	(29%)	(*17*%)	(**15**%)	(***14***%)
	14400s	up. Bound $\overline{\alpha}\downarrow$	17.54	11.15	***10.14***	*10.29*	***10.07***
		(% of gap $\downarrow$)	(84%)	(17%)	(***7***%)	(*8*%)	(***6***%)
FMNIST-Conv LB $\underline{\alpha}$= 22.21	1000s	up. Bound $\overline{\alpha}\downarrow$	41.52	26.61	***23.27***	*24.09*	***22.78***
		(% of gap $\downarrow$)	(93%)	(20%)	(***5***%)	(*10*%)	(***3***%)
	14400s	up. Bound $\overline{\alpha}\downarrow$	37.26	24.06	***22.42***	*23.80*	***22.34***
		(% of gap $\downarrow$)	(72%)	(9%)	(***1***%)	(*9*%)	(***0.6***%)
CIFAR-Conv LB $\underline{\alpha}$= 19.04	1000s	up. Bound $\overline{\alpha}\downarrow$	36.57*	27.15	*25.94*	***24.52***	***23.29***
		(% of gap $\downarrow$)	(55*%)	(28%)	(*24*%)	(***20***%)	(***15***%)
	14400s	up. Bound $\overline{\alpha}\downarrow$	31.53*	22.73	***21.77***	*21.81*	***20.38***
		(% of gap $\downarrow$)	(39*%)	(12%)	(***10***%)	(***10***%)	(***5***%)

Table 1: Upper bound $\bar{\alpha}$ and (gap to lower bound LB $\underline{\alpha}$), lower is better, for $L_\infty = 0.05$ perturbation, averaging bounds $\bar{\alpha}_{C,D}$ over all pair of classes (C, D). *Best* reports the average $\bar{\alpha}$ using the lowest $\bar{\alpha}_{C,D}$ computed in parallel by *Diff* and *2b+1* for each (C, D). Lower gaps using *Best* means that there are cases (C, D) where *Diff* is much better than *2b+1*, and cases where it is the opposite. * means that VHAGaR failed to compute bounds for some cases (C, D) of CIFAR-Conv. For such cases, we used numbers from ITNE instead (which is overperforming VHAGaR in every test), so these numbers are an optimistic evaluation for VHAGaR.

implemented. We tested with 1000s and 14400s (4h) timeout. We report the upper bound $\bar{\alpha}$ each MILP model finds. For each DNN, we report the best lower bound $\underline{\alpha}$ among the MILP models, and report the gap between $\bar{\alpha}$ and $\underline{\alpha}$ as $\%$ of increase over $\underline{\alpha}$. We report results in Table 1. Recall that most robustness tools, e.g., α, β-Crown, can only handle *local* robustness and cannot be tested.

Discussion: This test is relatively easy, where the best upper bound $\bar{\alpha}$ is close to the lower bound $\underline{\alpha}$, with at most a 6% gap. Using the *1b* model is not useful, as its abstraction will lose more than 6%. VHAGaR computes upper bounds $\bar{\alpha}$ quite far from the lower bound, with a gap $40-120\%$. Although we are using its benchmarks, VHAGaR methodology is not well-suited for L_∞ (it is more efficient on *Patch* Kabaha & Cohen (2024)): ITNE is better, with gaps from $9-17\%$. Our method provides the *Best* overall gaps: $0.6-6\%$, significantly tighter. Breaking up numbers, *2b+1* is more accurate for shorter timeout, and *Diff* is more accurate when there is sufficient time, as expected. Anyway, both are better for particular classes (C, D), as *Best* is lower than both *Diff* and *2b+1* in every test.

6.2 Comparison between LAE-DNNs and DNNs for L_1 perturbations.

We then turn to testing with L_1 perturbations and comparing between DNNs and their LAE-DNN versions (which are not supported in VHAGaR). Here, we consider MNIST-FC, which has a higher accuracy than MNIST-Conv, to test a different architecture than convolutional Networks.

LAE-DNNs are generated as described in Section 5: for each latent space dimension, we report in Table 2 the accuracy loss from using LAE-DNNs vs using the Vanilla DNNs. Concerning the LAE algorithm, we experimented with PCA (Chinesta et al., 2017). We set the dimension at the minimal dimension that results in at most 1% loss of accuracy, that is $20, 25$ and 60 for MNIST-FC, FMNIST-Conv and CIFAR10-Conv respectively, i.e. $>$ 30x dimension reduction in every case.

latent space dimension:	15	20	25	30	35	40	...	60
LAE-MNIST-FC	2%	0%	0%	0%	0%	0%		0%
LAE-FMNIST-Conv	5%	3%	1%	0%	0%	0%		0%
LAE-CIFAR10-Conv	16%	14%	12%	10%	14%	5%		1%

Table 2: Loss of accuracy of LAE-DNN vs DNN according to the latent space dimension. Blue corresponds to the minimal dimension ensuring loss $\leq 1\%$.

We consider the $\bar{\beta}$ bounds reached for L_1-perturbation at most 1, on the 3 original DNNs MNIST-FC, FMNIST-Conv, and CIFAR10-Conv, plus their LAE variants for the dimension computed in Table 2. We report the results in Table 3, in the same format as Table 1.

Discussion: Compared with L_∞ (Table 1), the gap between the upper bound $\bar{\beta}$ and the lower bound $\underline{\beta}$ is much larger with L_1 perturbations, from 1.5x to 600x. Using the faster *1b* model now becomes

$L_1 = 1$	Timeout	ITNE	*Diff*	*2b+1*	*1b*	*Best*
MNIST-FC	1000s	45.6	38.88	38.68	**37.34**	**37.33**
LB $\underline{\beta}$= 0.50	14400s	45.3	37.24	36.96	**33.68**	**33.67**
LAE-MNIST-FC	1000s	2.52	**2.43**	**2.43**	2.68	**2.41**
LB $\underline{\beta}$= 0.12	14400s	2.33	2.18	**2.00**	2.46	**2.00**
FMNIST-Conv	1000s	12.27	8.90	**7.55**	9.84	**7.55**
LB $\underline{\beta} = 4.16$	14400s	11.03	7.43	**6.37**	9.62	**6.22**
LAE-FMNIST-Conv	1000s	3.64	3.02	3.02	**2.74**	**2.74**
LB $\underline{\beta} = 0.96$	14400s	2.67	2.34	**1.95**	2.52	**1.95**
CIFAR10-Conv	1000s	10.49	8.30	**8.14**	8.17	**7.77**
LB $\underline{\beta} = 3.38$	14400s	10.49	**7.19**	7.45	7.31	**6.67**
LAE-CIFAR10-Conv	1000s	1.70	1.63	1.54	**1.28**	**1.28**
LB $\underline{\beta}$= 0.0015	14400s	1.67	1.56	1.52	**1.15**	**1.15**
extra long timeout	216000s	1.66	1.53	1.36	**0.99**	**0.99**

Table 3: Comparison of upper bound $\overline{\beta}$ (lower is better $\downarrow$) averaged over all bounds $\overline{\beta}_{C,D}$, between ITNE and *Best* of *Diff, 2b+1 and 1b*, for DNNs and LAE-DNNs over MNIST, FMNIST and CIFAR10.

pertinent. In half of the cases, *1b* actually produces the best results. When the timeout is large enough compared to the complexity of the model, $2b{+}1$ and sometimes *Diff* provide better bounds. ITNE is significantly worse, producing 25% worse bounds $\bar{\beta}$ on average than *Best*.

LAE-DNNs reach much smaller upper bounds $\bar{\beta}$ than DNN: 15x for MNIST-FC, 3x for FMNIST-Conv, and 7x for CIFAR10-Conv. LAE-CIFAR10-Conv is the hardest test, because the latent space dimension (60) is not as small as for MNIST and FMNIST (20 and 25), which is due to the fact that the original CIFAR-10 dimensions (3072) is 4x larger than for MNIST (784). Further, the *upper* bound $\bar{\beta}$ for LAE-CIFAR10-Conv (0.99) is much smaller than the *lower* bound $\underline{\beta}$ for CIFAR10-Conv (3.38), meaning that no other method would reach for CIFAR10-Conv a bound less than 3 times as high as the bound of our MILP models for LAE-CIFAR10-Conv. For FMNIST-Conv, it is 2x.

6.3 Real-Time Verification of Robustness from $(\bar{\beta}^{\varepsilon}_{C,D})_{C<D\leq 10}$

We now turn to how many images can be certified robust in real-time using the *Best* bounds $\beta^1_{C,D}$ computed in the previous subsection. We report the $\%$ of images certified robust in Table 4, as well as the latency overhead to certify them: it is half a millisecond in every case (using a single CPU core), as it depends only on the number of outputs of the DNN, which is 10 for all 6 DNNs considered. It means 2000 images can be treated per second by each CPU core. The complexity being linear with the number of classes, a dataset with 100 classes would need $\approx$ 5ms to be treated, 200 images/second per CPU core, which can be deemed *real-time*. We set a threshold of 70% to evaluate how large the L_1 perturbation can be while still certifying at least this threshold: a threshold much lower would mean labeling too many incoming images as uncertified, which would degrade the system too much.

Benchmark	$L_1 = 1$	$L_1 = 2$	$L_1 = 3$	$L_1 = 4$	Latency overhead
MNIST-FC	0%	0%	0%	0%	0.5 ms
LAE-MNIST-FC	80%	21%	1%	0%	0.5 ms
FMNIST-Conv	75%	50%	28%	12%	0.5 ms
LAE-FMNIST-Conv	92%	86%	79%	70%	0.5 ms
CIFAR10-Conv	39%	0%	0%	0%	0.5 ms
LAE-CIFAR10-Conv	92%	72%	61%	51%	0.5 ms

Table 4: Percentage of images certified robust in real-time using the Best computed $(\beta^1_{C,D})_{C<D\leq 10}$, for different values of L_1-perturbations. Blue depicts perturbation ensuring $\geq 70\%$ of images being robust. The latency overhead is the same as the number of classes (10) for all datasets.

With that, we can certify at least 70% of the images for all 3 LAE-DNNs for a L_1-perturbation of 1, and even up to perturbation 2 for CIFAR10 and even 4 for FMNIST. We are unable to do it for the non-LAE version, but in the case of FMNIST-Conv, with a perturbation 4 times smaller than for its LAE-version. Overall, LAE-DNNs are amenable to certification in real-time, unlike vanilla DNNs.

7 Conclusion

In this paper, we tackled *real-time* (few ms) certification for global robustness. The main idea is to proceed in 2 steps: first, compute *once, offline* global robustness bounds $\bar{\beta}$ on the DNN; then, use a simple look-up table based on $\bar{\beta}$ to perform the *online* certification. On vanilla DNN architectures, global bounds are too large ($\gg 4.16$ for FMNIST-Conv, 3.38 for CIFAR10-Conv, Table 3): real-time certification is impossible, irrespective of the method used to compute the bounds. To overcome this, we developed 2 complementary techniques that together yield offline bounds $\bar{\beta}$ good enough to certify $> 70\%$ of images. First, the LAE-DNN architecture retains high accuracy and avoids the very large β associated with non-ID inputs (see Fig. 1): the $\bar{\beta}$ computed are 3–15x lower. Second, our new *Diff* MILP method, which is more accurate for global robustness than the classical MILP encoding, yields, on average, 25% lower bounds than the state-of-the-art ITNE model (Wang et al., 2023).

This is a concept and feasibility paper, showing the feasibility of real-time robustness certification with limited accuracy loss. Many implementation details are out of the scope of the paper: testing other (L)AEs, non-MILP techniques, more activation functions, larger DNNs, etc. These are left for future work, as there is no theoretical limitations to our two-step technique.

AI USE STATEMENT

In this work, we used generative AI tools for text editing. We have not used generative AI tools for any other tasks with required disclosure, and all other required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for text editing to improve readability and grammar. We have reviewed all AI-assisted work by manually checking the edited text against the original content to ensure the underlying meaning, technical accuracy, and claims remained intact. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The proposed work does not involve human subjects, crowdsourcing, or sensitive data, and therefore does not raise any of the ethical concerns described in the guidelines.

REPRODUCIBILITY STATEMENT

The DNNs are openly available, as well as the datasets. The algorithms are clearly provided, and results can be replicated using Gurobi as solver using one's preferred code base. We will release the code upon acceptance of the paper.

REFERENCES

Stanley Bak. nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement. In *NASA Formal Methods Symposium*, pp. 19–36. Springer, 2021.

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring Neural Net Robustness with Constraints. In *NeurIPS 2016*, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/980ecd059122ce2e50136bda65c25e07-Abstract.html.

Christopher Brix, Mark Niklas Müller, Stanley Bak, Taylor T. Johnson, and Changliu Liu. First three years of the international verification of neural networks competition (vnn-comp), 2023.

Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020.

Yizheng Chen, Shiqi Wang, Yue Qin, Xiaojing Liao, Suman Jana, and David A. Wagner. Learning Security Classifiers with Verified Global Robustness Properties. In *CCS 2021*, 2021. URL https://doi.org/10.1145/3460120.3484776.

Francisco Chinesta, Antonio Huerta, Gianluigi Rozza, and Karen Willcox. *Model Reduction Methods*, pp. 1–36. John Wiley & Sons, 2017. doi: 10.1002/9781119176817.ecm2110.

Eduardo De Conto, Blaise Genest, and Arvind Easwaran. Function+Data Flow: A Framework to Specify Machine Learning Pipelines for Digital Twinning. In Bram Adams, Thomas Zimmermann, Ipek Ozkaya, Dayi Lin, and Jie M Zhang (eds.), *AIware'24*, pp. 19–27. ACM, 2024.

Serge Durand, Augustin Lemesle, Zakaria Chihani, Caterina Urban, and Francois Terrier. Reciph: Relational coefficients for input partitioning heuristic. In *1st Workshop on Formal Verification of Machine Learning (WFVML 2022)*, 2022. URL https://pyrat-analyzer.com/.

Claudio Ferrari, Mark Niklas Mueller, Nikola Jovanović, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations (ICLR'22)*, 2022.

Feisi Fu, Zhilu Wang, Weichao Zhou, Yixuan Wang, Jiameng Fan, Chao Huang, Qi Zhu, Xin Chen, and Wenchao Li. REGLO: Provable Neural Network Repair for Global Robustness Properties. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of*

Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pp. 12061–12071. AAAI Press, 2024. URL https://doi.org/10.1609/aaai.v38i11.29094.

Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark W. Barrett. DeepSafe: A Data-Driven Approach for Assessing Robustness of Neural Networks. In *ATVA 2018*, 2018. URL https://doi.org/10.1007/978-3-030-01090-4_1.

Chao Huang, Jiameng Fan, Xin Chen, Wenchao Li, and Qi Zhu. Divide and slide: Layer-wise refinement for output range analysis of deep neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3323–3335, 2020. doi: 10.1109/TCAD.2020.3013071.

Anan Kabaha and Dana Drachsler Cohen. Verification of Neural Networks' Global Robustness. *Proc. ACM Program. Lang.*, 8(OOPSLA1), 2024. doi: 10.1145/3649847. URL https://doi.org/10.1145/3649847.

Naveen Karunanayake, Ravin Gunawardena, Suranga Seneviratne, and Sanjay Chawla. Out-of-Distribution Data: An Acquaintance of Adversarial Examples - A Survey. *ACM Comput. Surv.*, 57(8):210:1–210:40, 2025. doi: 10.1145/3719292. URL https://doi.org/10.1145/3719292.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In Rupak Majumdar and Viktor Kunčak (eds.), *Computer Aided Verification*, pp. 97–117, Cham, 2017. ISBN 978-3-319-63387-9.

Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In Isil Dillig and Serdar Tasiran (eds.), *Computer Aided Verification*, pp. 443–452, Cham, 2019. Springer International Publishing. ISBN 978-3-030-25540-4.

Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-Robust Neural Networks. In *ICML 2021*, 2021. URL http://proceedings.mlr.press/v139/leino21a.html.

Natan Levy, Raz Yerushalmi, and Guy Katz. gRoMA: a Tool for Measuring Deep Neural Networks Global Robustness. In *arXiv.2301.02288*, 2023. URL https://doi.org/10.48550/arXiv.2301.02288.

Ravi Mangal, Aditya V. Nori, and Alessandro Orso. Robustness of neural networks: a probabilistic and practical approach. In *ICSE 2019*, 2019. URL https://doi.org/10.1109/ICSE-NIER.2019.00032.

MOSEK ApS. *MOSEK Modeling Cookbook*, 2025. URL https://docs.mosek.com/modeling-cookbook/index.html. Release 3.4.0.

Mark Niklas Muller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations. In *Proc. ACM Program. Lang. POPL*, volume 6, New York, NY, USA, 2022. Association for Computing Machinery. doi: 10.1145/3498704.

Brandon Paulsen, Jingbo Wang, and Chao Wang. Reludiff: differential verification of deep neural networks. In *ICSE'20*, pp. 714–726. ACM, 2020.

Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, and Marta Kwiatkowska. Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance. In *IJCAI 2019*, 2019. URL https://doi.org/10.24963/IJCAI.2019/824.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An Abstract Domain for Certifying Neural Networks. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019. doi: 10.1145/3290354.

Sahil Singla and Soheil Feizi. Improved techniques for deterministic l2 robustness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16110–16124. Curran Associates, Inc., 2022. doi: 10.52202/068431-1172.

Weidi Sun, Yuteng Lu, Xiyue Zhang, and Meng Sun. DeepGlobal: A framework for global robustness verification of feedforward neural networks. *In J. Syst. Archit.*, 2022. URL https://doi.org/10.1016/J.SYSARC.2022.102582.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations (ICLR'14)*, 2014.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *International Conference on Learning Representations (ICLR'19)*, 2019.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29909–29921. Curran Associates, Inc., 2021.

Zhilu Wang, Chao Huang, and Qi Zhu. Efficient global robustness certification of neural networks via interleaving twin-network encoding. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1087–1092. IEEE, 2022a.

Zhilu Wang, Yixuan Wang, Feisi Fu, Ruochen Jiao, Chao Huang, Wenchao Li, and Qi Zhu. A Tool for Neural Network Global Robustness Certification and Training. In *arXiv 2208.07289*, 2022b. URL https://doi.org/10.48550/arXiv.2208.07289.

Zhilu Wang, Chao Huang, and Qi Zhu. Efficient Global Robustness Certification of Neural Networks via Interleaving Twin-Network Encoding (Extended Abstract). In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pp. 6498–6503. ijcai.org, 2023. doi: 10.24963/IJCAI.2023/727. URL https://doi.org/10.24963/ijcai.2023/727.

Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark Barrett. Marabou 2.0: A versatile formal analyzer of neural networks. In Arie Gurfinkel and Vijay Ganesh (eds.), *Proceedings of the 36th International Conference on Computer Aided Verification (CAV '24)*, volume 14681 of *Lecture Notes in Computer Science*, pp. 249–264. Springer, July 2024. Montreal, Canada.

Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations (ICLR'21)*. OpenReview.net, 2021.

Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Rethinking Lipschitz Neural Networks and Certified Robustness: A Boolean Function Perspective. In *NeurIPS 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/7b04ec5f2b89d7f601382c422dfe07af-Abstract-Conference.html.

APPENDICES

A APPENDIX FOR SECTION 2: L_1-PERTURBATIONS

Proposition 2. *The following system of LP constraints is equivalent with* $\sum_{j\le k}|x_j - x'_j| \le \varepsilon$ *(L_1-perturbation):*

1. *for all* input *neuron* j *(in layer 0),* $A_j \ge x_j - x'_j$ *and* $A_j \ge x'_j - x_j$,
2. $\sum_{j\le k} A_j \le \varepsilon$.

Proof. First implication: For any solution to the constraint $\sum_{j\le k}|x_j - x'_j| \le \varepsilon$, we show that it also satisfies the combined constraints. We let $A_j = |x_j - x'_j|$ for each j. For each j, $A_j = |x_j - x'_j| \ge x_j - x'_j$, $A_j = |x_j - x'_j| \ge x'_j - x_j$, and

$$\sum_{j\le k} A_j = \sum_{j\le k} |x_j - x'_j| \le \varepsilon.$$

Reverse implication: Similarly, if A_j satisfies the system of LP constraints, then for any j, we have

$$A_j \ge \max\{x'_j - x_j, x_j - x'_j\} = |x_j - x'_j|.$$

Hence

$$\sum_{j\le k} |x_j - x'_j| \le \sum_{j\le k} A_j \le \varepsilon.$$

□

B APPENDIX FOR SECTION 3: REAL-TIME CERTIFICATION

Proposition 3. *If "certified robust" is returned over input* I*, then for all perturbed input* I' *with* $|I - I'| \le \varepsilon$*, the DNN has the same decision on* I *and on* I'*, i.e. the DNN is robust around* I*.*

Proof. By definition, "certified robust" means that for any $D \ne C$, $b_D = 0$, which means that $\bar{\beta}^{\varepsilon}_{C,D} < x_C - x_D$.

Then by the definition, for the image I' that $|I - I'| \le \varepsilon$, we have $(x_C - x_D) - (x'_C - x'_D) \le \bar{\beta}^{\varepsilon}_{C,D}$. Hence, $x'_C - x'_D \ge (x_C - x_D) - \bar{\beta}^{\varepsilon}_{C,D} > 0$. This argument works for all $D \ne C$, therefore, the output for I' is still C. □

We now compare our bound β with the bound α considered by Vhagar Kabaha & Cohen (2024), for an evaluation purpose only (how narrow is the gap) rather than to verify robustness in real-time. Namely, VHAGaR Kabaha & Cohen (2024) defines $\alpha^{\varepsilon}_{C,D}$, computing the perturbation necessary to switch an image classified as class C into an image classified as class D:

$$\alpha^{\varepsilon}_{C,D} = \max_{|I-I'|\le\varepsilon} (x_C - \max_{E\ne C} x_E) \qquad \text{s.t. } x'_D \ge \max_{E\ne D} x'_E \wedge x_C \ge \max_{E\ne C} x_E$$

Notice that $\alpha^{\varepsilon}_{C,D}$ is not symmetrical, while $\beta^{\varepsilon}_{C,D}$ is.

C APPENDIX FOR SECTION 4: LAE-DNN

We show that an LAE-DNN can be represented with the same architecture as the original DNN by absorbing the linear transformation of LAE-encode and LAE-decode into the first linear layer of the DNN.

Let LAE-encode be represented by a matrix $\boldsymbol{W}^{\text{enc}} \in \mathbb{R}^{n\times d_0}$ and LAE-decode by a matrix $\boldsymbol{W}^{\text{dec}} \in \mathbb{R}^{d_0\times n}$. Hence, for an input $\boldsymbol{x} \in \mathbb{R}^{d_0}$, the projected input produced by the LAE is

$$\boldsymbol{W}^{\text{dec}}\boldsymbol{W}^{\text{enc}}\boldsymbol{x}.$$

Proposition. *Let f be an ℓ-layer DNN with weight matrices $\boldsymbol{W}^i$ and bias vectors $\boldsymbol{b}^i$, as defined in Section 2. The LAE-DNN can be represented by an ℓ-layer DNN with the same layer dimensions and ReLU activations as f, by changing only the weight matrix of its first layer.*

Proof. The pre-activation values of the first layer of the LAE-DNN are

$$\begin{aligned}\boldsymbol{z}^1_{\mathrm{LAE}} &= \boldsymbol{W}^1\left(\boldsymbol{W}^{\mathrm{dec}}\boldsymbol{W}^{\mathrm{enc}}\boldsymbol{x}\right) + \boldsymbol{b}^1 \\ &= \left(\boldsymbol{W}^1\boldsymbol{W}^{\mathrm{dec}}\boldsymbol{W}^{\mathrm{enc}}\right)\boldsymbol{x} + \boldsymbol{b}^1.\end{aligned}$$

Define

$$\widetilde{\boldsymbol{W}}^1 = \boldsymbol{W}^1\boldsymbol{W}^{\mathrm{dec}}\boldsymbol{W}^{\mathrm{enc}}.$$

We then have

$$\boldsymbol{z}^1_{\mathrm{LAE}} = \widetilde{\boldsymbol{W}}^1\boldsymbol{x} + \boldsymbol{b}^1.$$

Thus, the LAE transformation is absorbed into the weight matrix of the first layer, while its bias vector remains unchanged.

All subsequent weight matrices and bias vectors remain unchanged. Therefore, all subsequent pre-activation and post-activation values, and in particular the output $\boldsymbol{z}^\ell$, are identical.

Finally,

$$\widetilde{\boldsymbol{W}}^1 \in \mathbb{R}^{d_1 \times d_0},$$

which is the same dimension as $\boldsymbol{W}^1$. Hence, this transformation changes neither the number of layers nor their dimensions, and the resulting DNN computes exactly the same function as the LAE-DNN. □

D Appendix for Section 5: *Diff* and *1b* MILP models

We first prove that the *Diff* model encodes exactly $\hat{y}$.

Proposition 4. *Assuming $y_j \in [-\gamma_j, \gamma_j]$ and $x'_j, y_j + x'_j \in [\alpha_j, \beta_j]$, we have that $\hat{y}_j = \mathrm{ReLU}(x_j = y_j + x'_j) - \mathrm{ReLU}(x'_j)$ is the solution of the MILP program:*

$$\begin{aligned}
y_j + x'_j &\le a\beta_j & y_j + x'_j &\ge (1-a)\alpha_j \\
x'_j &\le a'\beta_j & x'_j &\ge (1-a')\alpha_j \\
\hat{y}_j &\le a\gamma_j & \hat{y}_j &\ge -a'\gamma_j \\
\hat{y}_j &\le y_j + x'_j - (1-a)\alpha_j & \hat{y}_j &\ge y_j + x'_j - a'\beta_j \\
\hat{y}_j &\le -x'_j + a\beta_j & \hat{y}_j &\ge -x'_j + (1-a')\alpha_j \\
\hat{y}_j &\le y_j + (1-a)\gamma_j & \hat{y}_j &\ge y_j - (1-a')\gamma_j
\end{aligned}$$

where $a, a' \in \{0,1\}$ are binary variables, and a' is shared with the classical MILP encoding for $\hat{x}'_j = \mathrm{ReLU}(x'_j)$.

Proof. First, we list the inequalities:

$$\begin{aligned}
y_j + x'_j &\le a\beta_j & y_j + x'_j &\ge (1-a)\alpha_j \\
x'_j &\le a'\beta_j & x'_j &\ge (1-a')\alpha_j \\
\hat{y}_j &\le a\gamma_j & \hat{y}_j &\ge -a'\gamma_j \\
\hat{y}_j &\le y_j + x'_j - (1-a)\alpha_j & \hat{y}_j &\ge y_j + x'_j - a'\beta_j \\
\hat{y}_j &\le -x'_j + a\beta_j & \hat{y}_j &\ge -x'_j + (1-a')\alpha_j \\
\hat{y}_j &\le y_j + (1-a)\gamma_j & \hat{y}_j &\ge y_j - (1-a')\gamma_j
\end{aligned}$$

Following the discussion in the proof of the main text, we need to check the other inequalities in each case.

$\mathbf{a} = \mathbf{0}, \mathbf{a}' = \mathbf{0}$**:** In this case, the inequalities become:

$$\begin{aligned} y_j + x'_j &\le 0 & y_j + x'_j &\ge \alpha_j \\ x'_j &\le 0 & x'_j &\ge \alpha_j \\ \hat{y}_j &\le 0 & \hat{y}_j &\ge 0 \\ \hat{y}_j &\le y_j + x'_j - \alpha_j & \hat{y}_j &\ge y_j + x'_j \\ \hat{y}_j &\le -x'_j & \hat{y}_j &\ge -x'_j + \alpha_j \\ \hat{y}_j &\le y_j + \gamma_j & \hat{y}_j &\ge y_j - \gamma_j \end{aligned}$$

Now we can see, $y_j + x'_j \le 0$ and $x'_j \le 0$ (lines 1&2), $\hat{y}_j = 0$ (line 3).

For the other 3 lines, by the above discussion and the assumption of the proposition that $y_j \in [-\gamma_j, \gamma_j]$ and $x'_j, y_j + x'_j \in [\alpha_j, \beta_j]$, they are implied by the following inequalities respectively (one for one)

$$\begin{aligned} \hat{y}_j &\le 0 \le y_j + x'_j - \alpha_j & \hat{y}_j &\ge 0 \ge y_j + x'_j \\ \hat{y}_j &\le 0 \le -x'_j & \hat{y}_j &\ge 0 \ge -x'_j + \alpha_j \\ \hat{y}_j &\le 0 \le y_j + \gamma_j & \hat{y}_j &\ge 0 \ge y_j - \gamma_j \end{aligned}$$

Therefore, all 3 rest lines are implied by other inequalities.

$\mathbf{a} = \mathbf{1}, \mathbf{a}' = \mathbf{0}$**:** In this case, the inequalities become:

$$\begin{aligned} y_j + x'_j &\le \beta_j & y_j + x'_j &\ge 0 \\ x'_j &\le 0 & x'_j &\ge \alpha_j \\ \hat{y}_j &\le \gamma_j & \hat{y}_j &\ge 0 \\ \hat{y}_j &\le y_j + x'_j & \hat{y}_j &\ge y_j + x'_j \\ \hat{y}_j &\le -x'_j + \beta_j & \hat{y}_j &\ge -x'_j + \alpha_j \\ \hat{y}_j &\le y_j & \hat{y}_j &\ge y_j - \gamma_j \end{aligned}$$

Now we can see, $y_j + x'_j \ge 0$ and $x'_j \le 0$ (lines 1&2), hence $y_j \ge 0$; and $\hat{y}_j = y_j + x'_j$ (line 4).

By the above discussion, we have:

$$\hat{y}_j \le y_j + x'_j \le y_j \quad \hat{y}_j \ge y_j + x'_j \ge 0$$

Then, by the assumption of the proposition that $y_j \in [-\gamma_j, \gamma_j]$ and $x'_j, y_j + x'_j \in [\alpha_j, \beta_j]$, all 3 other lines are implied by the following inequalities respectively (one for one):

$$\begin{aligned} \hat{y}_j &\le y_j \le \gamma_j & \hat{y}_j &\ge y_j + x'_j \ge 0 \\ \hat{y}_j &\le y_j + x'_j \le \beta_j \le -x'_j + \beta_j & \hat{y}_j &\ge 0 \ge -x'_j + \alpha_j \\ \hat{y}_j &\le y_j + x'_j \le y_j & \hat{y}_j &\ge 0 \ge y_j - \gamma_j \end{aligned}$$

Therefore, all 3 rest lines are implied by other inequalities.

$\mathbf{a} = \mathbf{0}, \mathbf{a}' = \mathbf{1}$**:** In this case, the inequalities become:

$$\begin{aligned} y_j + x'_j &\le 0 & y_j + x'_j &\ge \alpha_j \\ x'_j &\le \beta_j & x'_j &\ge 0 \\ \hat{y}_j &\le 0 & \hat{y}_j &\ge -\gamma_j \\ \hat{y}_j &\le y_j + x'_j - \alpha_j & \hat{y}_j &\ge y_j + x'_j - \beta_j \\ \hat{y}_j &\le -x'_j & \hat{y}_j &\ge -x'_j \\ \hat{y}_j &\le y_j + \gamma_j & \hat{y}_j &\ge y_j \end{aligned}$$

Now we can see, $y_j + x'_j \le 0$ and $x'_j \ge 0$ (lines 1&2), hence $y_j \le 0$; and $\hat{y}_j = -x'_j$ (line 5).

By the above discussion, we have:

$$\hat{y}_j \le -x'_j \le 0 \qquad \hat{y}_j \ge -x'_j \ge -x'_j + (y_j + x'_j) \ge y_j$$

Then, by the assumption of the proposition that $y_j \in [-\gamma_j, \gamma_j]$ and $x'_j, y_j + x'_j \in [\alpha_j, \beta_j]$, all 3 other lines are implied by the following inequalities, respectively (one for one):

$$\begin{aligned} &\hat{y}_j \leq -x'_j \leq 0 && \hat{y}_j \geq y_j \geq -\gamma_j \\ &\hat{y}_j \leq 0 \leq y_j + x'_j - \alpha_j && \hat{y}_j \geq y_j \geq y_j + x'_j - \beta_j \\ &\hat{y}_j \leq 0 \leq y_j + \gamma_j && \hat{y}_j \geq y_j \end{aligned}$$

Therefore, all 3 rest lines are implied by other inequalities.

$\mathbf{a} = \mathbf{1}, \mathbf{a}' = \mathbf{1}$**:** In this case, the inequalities become:

$$\begin{aligned} y_j + x'_j &\leq \beta_j & y_j + x'_j &\geq 0 \\ x'_j &\leq \beta_j & x'_j &\geq 0 \\ \hat{y}_j &\leq \gamma_j & \hat{y}_j &\geq -\gamma_j \\ \hat{y}_j &\leq y_j + x'_j & \hat{y}_j &\geq y_j + x'_j - \beta_j \\ \hat{y}_j &\leq -x'_j + \beta_j & \hat{y}_j &\geq -x'_j \\ \hat{y}_j &\leq y_j & \hat{y}_j &\geq y_j \end{aligned}$$

Now we can see, $y_j + x'_j \geq 0$ and $x'_j \geq 0$ (lines 1&2) and $\hat{y}_j = y_j$ (line 6).

Then, by the assumption of the proposition that $y_j \in [-\gamma_j, \gamma_j]$ and $x'_j, y_j + x'_j \in [\alpha_j, \beta_j]$, all 3 other lines are implied by the following inequalities respectively (one for one):

$$\begin{aligned} &\hat{y}_j \leq y_j \leq \gamma_j && \hat{y}_j \geq y_j \geq -\gamma_j \\ &\hat{y}_j \leq y_j \leq y_j + x'_j && \hat{y}_j \geq y_j \geq y_j + x'_j - \beta_j \\ &\hat{y}_j \leq -x'_j + (y_j + x'_j) \leq -x'_j + \beta_j && \hat{y}_j \geq y_j - (y_j + x'_j) \geq -x'_j \end{aligned}$$

Therefore, all 3 rest lines are implied by other inequalities.

□

We now turn to the *1b* MILP model. First, we provide in Fig. 4 the possible values $\hat{y}_i$ can take depending on y_i.

Proposition 5. *Given $y_i \in [-\gamma_i, \gamma_i]$, $\hat{y}_i$ is a solution of the MILP program :*

$$\begin{aligned} &\hat{y}_i \leq a\gamma_i && \hat{y}_i \geq y_i - a\gamma_i \\ &\hat{y}_i \geq (a-1)\gamma_i && \hat{y}_i \leq y_i + (1-a)\gamma_i \end{aligned}$$

where $a \in \{0, 1\}$ is a binary variable.

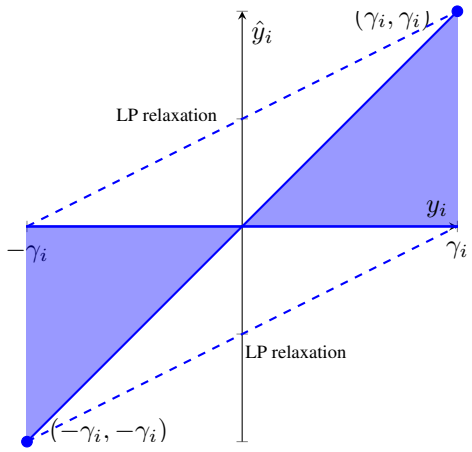

Figure 4: Possible values of $\hat{y}_i$ depending on y_i for the *1b* model.

Proof. First, we list the inequalities:

$$\hat{y}_i \leq a\gamma_i \qquad \hat{y}_i \geq y_i - a\gamma_i$$
$$\hat{y}_i \geq (a-1)\gamma_i \qquad \hat{y}_i \leq y_i + (1-a)\gamma_i$$

Similar to Proposition 4, we consider by cases.

$\mathbf{a = 0}$: In this case, the inequalities become:

$$\hat{y}_i \leq 0 \qquad \hat{y}_i \geq y_i$$
$$\hat{y}_i \geq -\gamma_i \qquad \hat{y}_i \leq y_i + \gamma_i$$

Now we can see, $y_i \leq 0$ (line 1) and $y_i \leq \hat{y}_i \leq 0$. Line 2 is implied by other inequalities since

$$\hat{y}_i \geq y_i \geq -\gamma_i \qquad \hat{y}_i \leq 0 \leq y_i + \gamma_i$$

$\mathbf{a = 1}$: In this case, the inequalities become:

$$\hat{y}_i \leq \gamma_i \qquad \hat{y}_i \geq y_i - \gamma_i$$
$$\hat{y}_i \geq 0 \qquad \hat{y}_i \leq y_i$$

Now we can see, $y_i \geq 0$ (line 2) and $0 \leq \hat{y}_i \leq y_i$. Line 1 is implied by other inequalities since

$$\hat{y}_i \leq y_i \leq \gamma_i \qquad \hat{y}_i \geq 0 \geq y_i - \gamma_i$$

Now we finish the proof. □

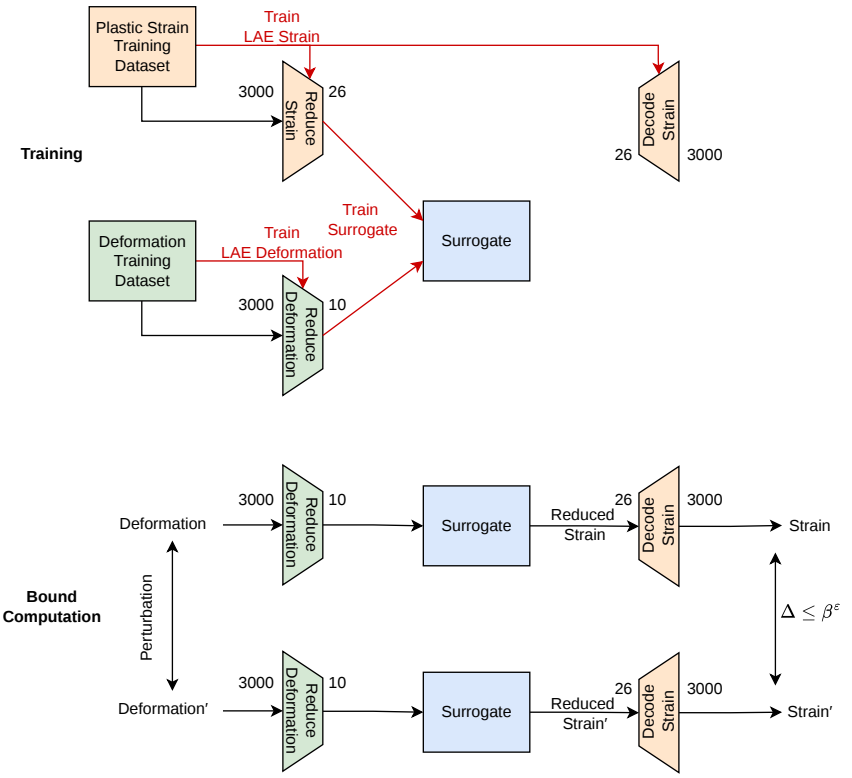

Figure 5: Training and bound computation on the pipe use case. Two LAE-encoders are used, for the input (deformation) and for the output space (plastic strain). The surrogate is learned from reduced deformation input to reduced strain output.

E APPENDIX FOR SECTION 6: *Diff* FOR ROBUSTNESS IN REGRESSION TASKS

Our *Diff* model is not only accurate for classification. It can also provide upper bounds on the output of the DNN-regressor due to small perturbations of the input. We consider here a pipe system and its DNN regressor from De Conto et al. (2024), that predicts the plastic strain in 3000 points of a mesh over a pipe given the deformation of each of these 3000 points. Compared to the classification tasks where the DNN is learned in a standard way and LAE is later applied to reduce the input dimension without compromising the accuracy, the DNN regressor has been learned on the reduced LAE feature space directly, for both the high-dimensional input and the high-dimensional output (3000 dimensions each).

Figure 5 describes the pipeline used to learn the DNN regressor for the pipe system. The DNN that has been learned has 2 fully connected layers, 50 neurons each, learned as a surrogate model, with 10 reduced inputs and 26 reduced outputs.

In terms of perturbation, we consider a conjunction of $L_1 \leq 3.9$ and $L_\infty \leq .02$ perturbations, physically pertinent dimensions, and maximize the sum of the difference in output of 10 selected points in the mesh between the input and its deformation.

Pipe	Timeout	ITNE	*Diff*	*2b+1*	*1b*
	1000s	0.045	0.042	0.041	**0.036**
Surrogate	14400s	0.042	0.036	**0.035**	0.036
LB = .025	72000s	0.042	0.034	**0.033**	0.036

Table 5: Comparison of upper bound (lower is better ↓) between *Diff*, *2b+1*, *1b* with ITNE on the pipe system with timeouts of 1000s, 14400s and 72000s.

For the pipe system, we compare in Table 5 the different models ITNE, *Diff*, $2b + 1$,$1b$ to produce bounds on the sum of the difference of strain over 10 specific points of the mesh, for a physically relevant perturbation of the deformation. The bounds we found are quite accurate, with a best bound of .033 obtained by the $2b + 1$ model, slightly better than the bound .034 found within the same time by the *Diff* model, and better than the bound found .036 by the *"*$1b$*"* model, although this bound has been found 70 times faster due to the simpler model (*1b* converges after 1000s). ITNE is significantly worse than even the *"*$1b$*"* with 1000s of timeout, at .042 despite a very long timeout. The certified lower bound is not too far, at .025.

We did check that this worst-case found, displayed in Fig. 6 and which is not too far from the actual worst case that is known to be $< .033$, is coherent with the physical finite element model the DNN surrogate has been learned from. Hence, this brittleness of the surrogate is *not* a hallucination due to the learning process, but rather correctly reflects the physical system.

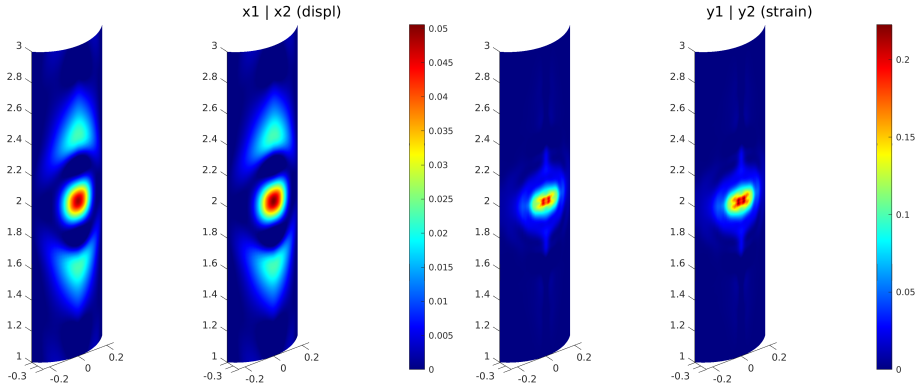

Figure 6: 2 slightly different deformations and their associated quite different strain as obtained by the *Diff* model with 200 binary variables.