Formal Safety Assessment of Neural Networks under Adversarial Attacks

Dore Mauro
mauro.dore@studio.unibo.it

Marongiu Gian Mario
gianmario.marongiu@studio.unibo.it

Murgia Riccardo
riccardo.murgia2@studio.unibo.it

May 26, 2025

Abstract

Providing formal guarantees on the behavior of machine learning models remains a major challenge for their deployment in safety-critical domains. The SMLE framework, recently introduced by Matteo Francobaldi and Michele Lombardi, proposes an architecture and training procedure that ensures compliance with designer-specified input-output properties through conservative overapproximation and counterexample-guided training [1]. This overapproximation is enforced by introducing two clipping parameters that scale the embedding produced by the feature extractor.
In this project we aim to extend SMLE's contributions by evaluating the framework's robustness and performance on a variety of settings. Our study involves training SMLE architectures and employing exact verification methods to systematically search for counterexamples, unlike the more relaxed verification approaches used in [1]. This study is conducted on a regression task using synthetic data. Additionally, we implement custom loss functions designed to regularize training. This comprehensive evaluation aims to assess the effectiveness of SMLE's verification mechanisms and to uncover insights that can inform the design of safer neural models with stronger formal guarantees. Ultimately, the tests performed as expected, demonstrating that a network trained with the SMLE approach can be a valuable tool in certain settings, effectively serving as a scalable method for the safe training of neural networks.

1 Introduction

Adversarial attacks are a significant challenge in neural networks. These attacks involve making small, often imperceptible changes to the input data that lead the model to produce incorrect outputs. In this context, the SMLE framework [1] proposes an efficient and scalable method to train neural networks that are safe against such attacks.
A neural network f can be viewed as the composition of a feature extractor h and a classification (or regression) head g, as illustrated in Figure 1. We assume that the input x to the network satisfies a set of properties P, and the output $f(x)$ satisfies another set of properties R. An adversarial attack, in this setting, consists of finding a counterexample, namely a perturbed input x such that $P(x)$ still holds, but $R(f(x))$ does not (eq. 1).

$$\exists x \mid P(x) \wedge \neg R(f(x)) \tag{1}$$

In order to claim that a neural network is absolutely safe from this type of adversarial attack, it is necessary to prove that no such x as defined in Equation 1 exists. This verification task can be formulated as an optimization problem; however, it becomes extremely computationally expensive to solve for large networks. Typically, the main bottleneck lies in the feature extractor h, which is usually the most resource-intensive component of the model.
In the SMLE framework, two additional layers are introduced to bound the embedding z produced by the feature extractor h. This updated architecture is illustrated in Figure 2. The exact verification

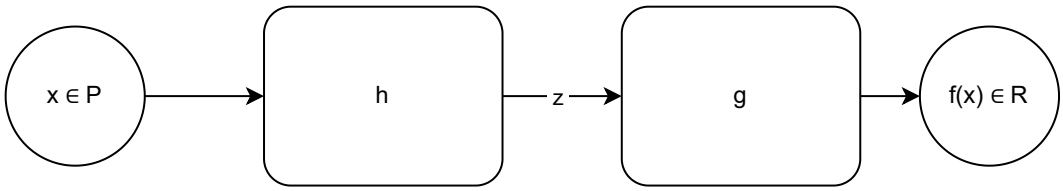

Figure 1: Simple neural network with feature extractor (h) and classification head (g)

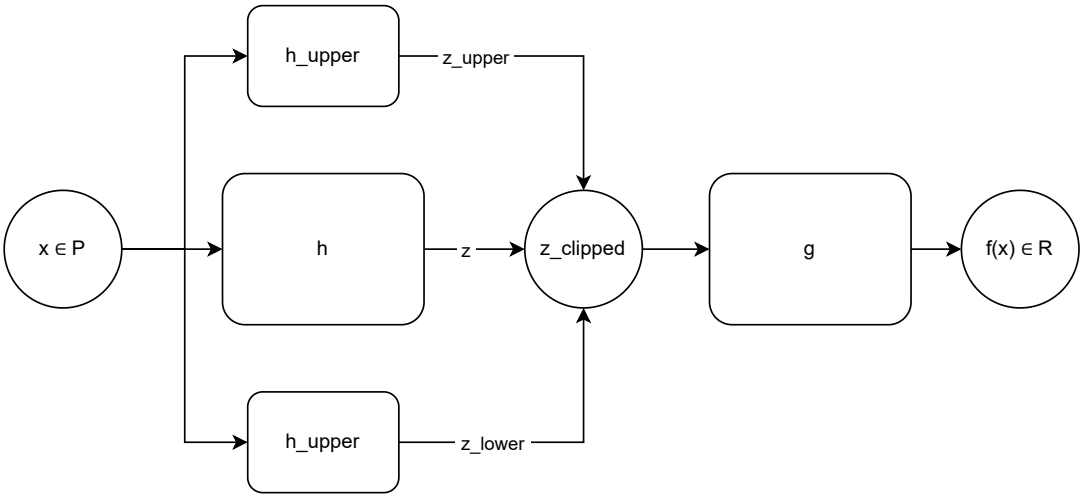

Figure 2: SMLE with feature extractor (h), layer for upper and lower bound (h_upper and h_lower) classification head (g)

problem for this network consists of finding an input x such that the embedding z lies within a constrained region defined by a clipping function applied to $h(x)$, while still satisfying the input property $P(x)$ and violating the output property $R(f(x))$, as expressed in 2.

$$\exists x \mid z_{clipped} = \text{clip}(h(x)) \land P(x) \land \neg R(f(x)) \tag{2}$$

However, even this setting can be computationally expensive. The key intuition behind the SMLE framework is that, at verification time, it is possible to discard the feature extractor h and perform the optimization over the remaining part of the network. By removing the feature extractor, the search for counterexamples is conducted over all possible embeddings z within the bounds defined by h_{lower} and h_{upper}:

$$\exists x, z \mid z \in \overline{\underline{H}} \land P(x) \land \neg R(f(x)) \tag{3}$$

where $\overline{\underline{H}}$ denotes the space defined by the clipping layers.

This formulation results in an overapproximation, which may lead to the detection of false positives—i.e., counterexamples that exist in the approximated verification setting (eq. 3) but not in the exact one (eq. 2). The main strength of this approach is its scalability, as the clipping layers are, by design, simple and efficient to verify. Moreover, if this test yields no counterexamples, we can confidently conclude that the network is safe. However, if counterexamples are found, the result is inconclusive, as they may correspond to false positives.
In the original paper, the idea is to use approximated verification during the model's training phase. At the end of each training step, if a violation is detected (i.e., a counterexample is found), a weight projection is performed to map the network's output back within the space defined by the constraints R.

In contrast, our approach consists of performing an unsafe training phase, followed by an external verification step. Specifically, we conduct the following studies:

- **Training of the unsafe model with six different complexity levels**, corresponding to all possible combinations of h and h_{aux} shown in Tables 1 and 2.
  Contributors: Gian Mario and Riccardo (training and architecture selection).
- **Implementation and evaluation of three loss functions** incorporating different regularization strategies to limit the size of the clipping box during training.
  Contributors: Riccardo (loss implementation), Gian Mario (training), Mauro (data collection and analysis).
- **Counterexample verification of three types**: approximated, exact, and exact with auxiliary clipping layers removed.
  Contributors: Gian Mario, Mauro, and Riccardo (optimization method implementation).
- **Analysis and comparison of the execution times** required by the three verification methods.
  Contributors: Mauro and Gian Mario.
- **Study of the termination conditions** of the verification methods.
  Contributor: Mauro.
- **Evaluation of false positives** (i.e., how often approximated verification finds a counterexample that is not an actual violation but rather a result of overapproximation).
  Contributors: Mauro and Riccardo.

It is important to note that we did not work in isolated compartments; the division of contributors is only approximate. Collaboration was frequent, with team members often seeking each other's help, so no task was carried out entirely in solitude.

2 Experimental Setup

2.1 Synthetic Regression Data

The synthetic regression dataset used in this study was originally introduced in the SMLE work [1]. In this setting, training data is generated in a controlled way to reflect a specific set of constraints. Input vectors $x \in R^d$ are sampled uniformly from the hypercube $[-1, 1]^d$, and the corresponding true labels $y \in R^2$ are computed by first summing the components of x, then applying a non-linear transformation based on powers of this sum. Specifically, for each sample, the labels are computed as in eq. 4.

$$y = \left(s^3, s^4\right), \quad \text{where} \quad s = \sum_{i=1}^{d} x_i. \tag{4}$$

The resulting output vectors are then standardized using the mean and standard deviation computed from the training set. In addition to the regression targets, both input and output vectors are required to satisfy a set of linear inequality constraints. These are defined as polytope membership tests, as in eq. 5.

$$P(x) \equiv Px \leq p \qquad R(f(x)) \equiv Rf(x) \leq r \tag{5}$$

Here, P and R are fixed coefficient matrices, and p, r are the corresponding right-hand side vectors. These constraints define valid regions in input and output spaces, ensuring that only samples within well-defined polytopes are considered during training and evaluation.

2.2 Exact Verification Formulation

We formulated the exact verification described in eq. 2 as a MILP optimization problem. The neural network was encoded in a format suitable for optimization using the `omlt` library [2]. In addition to the constraints that define the path of the input x through the network, we defined the following constraints to model the clipping operation using the big-M formulation:

Complexity	Architecture
1	8 [linear]
2	8 [relu] $\rightarrow$ 16 [relu] $\rightarrow$ 8 [linear]
3	8 [relu] $\rightarrow$ 16 [relu] $\rightarrow$ 24 [relu] $\rightarrow$ 16 [relu] $\rightarrow$ 8 [linear]

Table 1: Architectures tested as feature extractors h

- Binary variables: b_{low} indicates that the embedding z produced by h is lower than the lower bound z_{lower} produced by h_{lower}, while b_{up} indicates the opposite.

$$b_{\text{low}} = \begin{cases} 1 & \text{if } z \leq z_{\text{lower}} \\ 0 & \text{otherwise} \end{cases}, \quad b_{\text{up}} = \begin{cases} 1 & \text{if } z \geq z_{\text{upper}} \\ 0 & \text{otherwise} \end{cases}$$

- At most one bound can be active:

$$b_{\text{low}} + b_{\text{up}} \leq 1$$

- Clipped output must lie within bounds defined by the clipping layers:

$$z_{\text{clipped}} \geq z_{\text{lower}}$$

$$z_{\text{clipped}} \leq (1 - t) \cdot z_{\text{lower}} + t \cdot z_{\text{upper}}$$

  t here is an auxiliary binary variable used to handle the case where $z_{lower} > z_{upper}$.

- if $z \leq z_{lower}$ the lower bound for $z_{clipped}$ is z

$$z_{\text{clipped}} \leq z + M \cdot b_{\text{low}}$$

- if $z \geq z_{upper}$ the upper bound for $z_{clipped}$ is z

$$z_{\text{clipped}} \geq z - M \cdot b_{\text{up}}$$

- if $z \leq z_{lower}$ the upper bound for $z_{clipped}$ is z_{lower}

$$z_{\text{clipped}} \leq z_{\text{lower}} + M \cdot (1 - b_{\text{low}})$$

- if $z \geq z_{upper}$ the lower bound for $z_{clipped}$ is z_{upper}

$$z_{\text{clipped}} \geq z_{\text{upper}} - M \cdot (1 - b_{\text{up}})$$

Here, $M = 1e6$ is an arbitrarily large positive number that is used to conditionally enforce constraints depending on the values of the binary variables. It ensures that the correct clipping bounds are applied when the corresponding condition is active, while relaxing the constraint otherwise.

2.3 Networks and Loss Functions

We evaluate all combinations of feature extractor h and clipping layers $h_{\text{lower}}, h_{\text{upper}}$ across varying complexities. The architectures for h span three levels of increasing depth (Table 1), while the clipping layers are tested in both linear and non-linear forms (Table 2). This setup allows us to analyze how architectural choices affect both model behavior and verification performance. These architectures are trained with four different loss functions.

The first one is a simple Maximum Squared Error (MSE):

$$\mathbb{L}(x, y) = E_{x,y}\left[MSE(y, f(x))\right] \tag{6}$$

Complexity	Architecture
linear	8 [linear]
non-linear	8 [linear] → 3 [relu] → 8 [linear]

Table 2: Architectures tested as clipping layers h_{lower} and h_{upper}

The others include regularization terms designed to reduce the size of the clipping box during training. The loss $\mathbb{L}_1$ adds a penalty on the difference between the upper and lower clipping bounds, normalized by their respective sizes:

$$\mathbb{L}_1(x, y) = E_{x,y}\left[MSE(y, f(x)) + \lambda \frac{m}{k} \frac{\|h_{upper}(x) - h_{lower}(x)\|_1}{\|\theta_g\|_1}\right] \tag{7}$$

where λ is an arbitrarily small constant, k is the embedding dimension, and m is the output dimension of the network.

To avoid issues with non-zero gradients and instability in training due to the L_1 norm, eq. 8 replaces the L_1 terms with squared L_2 norms:

$$\mathbb{L}_2(x, y) = E_{x,y}\left[MSE(y, f(x)) + \lambda \frac{m}{k} \frac{\|h_{upper}(x) - h_{lower}(x)\|_2^2}{\|\theta_g\|_2^2}\right] \tag{8}$$

Finally, the fourth loss function separates the regularization into two independent terms with distinct hyperparameters λ' and λ'':

$$\mathbb{L}_3(x, y) = E_{x,y}\left[MSE(y, f(x)) + \lambda' \frac{1}{k}\|h_{upper}(x) - h_{lower}(x)\|_2^2\right] + \lambda'' \frac{1}{m}\|\theta_g\|_2^2 \tag{9}$$

where λ' and λ'' are constants.
This formulation allows more flexibility in tuning the trade-off between shrinking the clipping box and preserving representational capacity, at the cost of introducing an additional hyperparameter and potential scaling issues.

3 Results

3.1 Training

Training behaved as expected, with the loss decreasing in a similar manner across all the configurations presented. Moreover, the regularization term in the loss functions affected the clipping box magnitudes as desired, as shown in Fig. 3.

Looking also at the validation loss tracking in Fig. 4, there does not appear to be a clearly preferable configuration or training method. Each loss function leads to similar results over the full training process.

3.2 False Positives Analysis

A counterexample x found using the approximated method is considered valid if, when fed to the network in Figure 2, it produces an output $f(x) \notin R$. Conversely, x is a false positive if the resulting output satisfies the constraints, i.e., $f(x) \in R$.
In this context, the false positive rate exhibits significant variability depending on the intrinsic complexity of the model, without showing any consistent trend. This is due to the fact that training is performed in an unsafe manner, i.e., without weight projection when a violation is detected at each training step. As shown in Table 3, training an SMLE network without regularization does not result in significantly high false positive rates; however, when regularization is included, their number drops considerably. This is because a smaller clipping box leads to a more contained overapproximation, potentially limiting the regions where the output "leaks" outside the valid output property space.

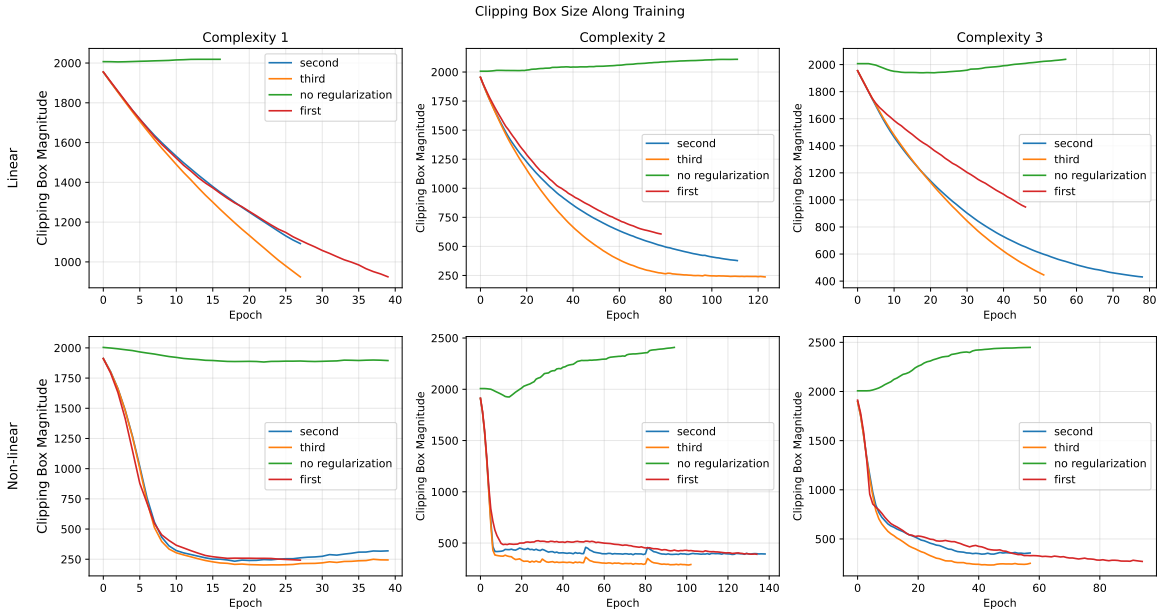

Figure 3: Clipping box magnitudes along training. "No regularization" refers to eq. 6; "first" refers to eq. 7; "second" refers to eq. 8; "third" refers to eq. 9

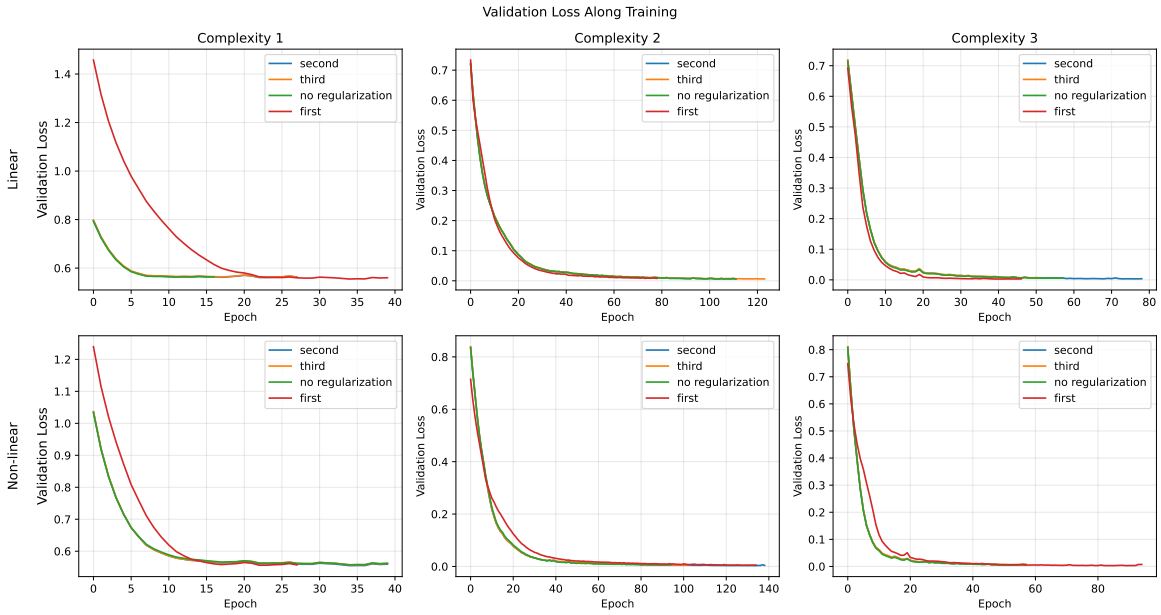

Figure 4: Validation loss along training. "No regularization" refers to eq. 6; "first" refers to eq. 7; "second" refers to eq. 8; "third" refers to eq. 9

3.3 Termination Conditions

Across all levels of complexity, models with linear clipping layers h_{lower} and h_{upper} consistently terminate with the optimal condition for both the Approximated (eq. 3) and Exact (eq. 2) verification methods.

In contrast, a more varied pattern is observed in models with non-linear clipping layers. While the Approximated method often succeeds in finding optimal solutions, the Exact method frequently encounters difficulties due to computational constraints, especially as complexity increases. This often leads to a `MaxTimeLimit` termination condition, where a sub-optimal solution is found, or an `aborted`

Complexity	MSE	MSE First	MSE Second	MSE Third
1	15%	0%	0%	0%
2	15%	5%	5%	5%
3	15%	5%	0%	0%
average	15%	3.33%	1.67%	1.67%

Table 3: Percentage of false positives per training loss function

condition, where the optimizer is unable to complete even the pre-solving phase within the 300-second time limit.
When performing exact verification on the simple network—specifically, counterexample search using eq. 1 on network 1, only the network with complexity 1 consistently terminated with an optimal solution. In 100% of cases for complexities 2 and 3, the optimization problem terminated with a `MaxTimeLimit` condition. This suggests that, despite the simplicity of the network (i.e., the absence of clipping layers), the clipping operations present in the SMLE structure aid the optimization process by pruning irrelevant regions of the search space.

3.4 Time Analysis

One of the main advantages of the approximated verification method is its relatively fast solving time when using a MILP optimizer. In our setting, we compared how much faster this method is compared to two other approaches:

1. Exact verification, where the counterexample search is performed on a network like the one shown in Figure 2.

2. Exact verification, with the search performed as in eq. 1 on a simpler network like the one in Figure 1. This network, which does not include clipping layers, corresponds to the standard feature extractor plus classification/regression head commonly used in classification tasks.

Verification time is measured across different model configurations and phases, as illustrated in Figure 5.

Verification times vary significantly depending on model complexity and the nature of the clipping layers. In general, exact verification is consistently more time-intensive than approximated verification, especially when the clipping layers are non-linear.
Architectures with linear clipping layers maintain low and stable verification times even as complexity increases, whereas non-linear clipping layers introduce a higher computational cost and scale poorly with added complexity. Additionally, the reduced number of valid samples with non-linear clipping layers at higher complexities limits the generalizability of those results.
An interesting observation arises in the case of the verification on the network shown in Figure 1. Despite the absence of clipping layers (and therefore fewer parameters and lower overall complexity), the only configuration that did not reach the `MaxTimeLimit` was the model with complexity 1. For complexities 2 and 3, it was not possible to collect sufficient data to compute meaningful statistics on verification time, as the optimization problem failed to find a solution within the time limit.
Overall, the data highlights that model complexity—and particularly the non-linearity in the clipping layers—has a disproportionate impact on verification time.

3.5 Counterexamples Comparison

One of the key performance indicators evaluated in this study is the number of times the approximated verification finds a counterexample x_{approx} different to the x_{exact} found through exact verification. This metric serves as a measure of the fidelity of the approximate solver in reproducing optimal or near-optimal solutions under different conditions of model complexity.

The data does not reveal any clear pattern across model complexities, except for the observation that higher complexity models exhibit a greater number of non-comparable instances. x_{approx} and x_{exact} are considered non-comparable when at least one of them is undefined due to the solver terminating with an `aborted` condition.

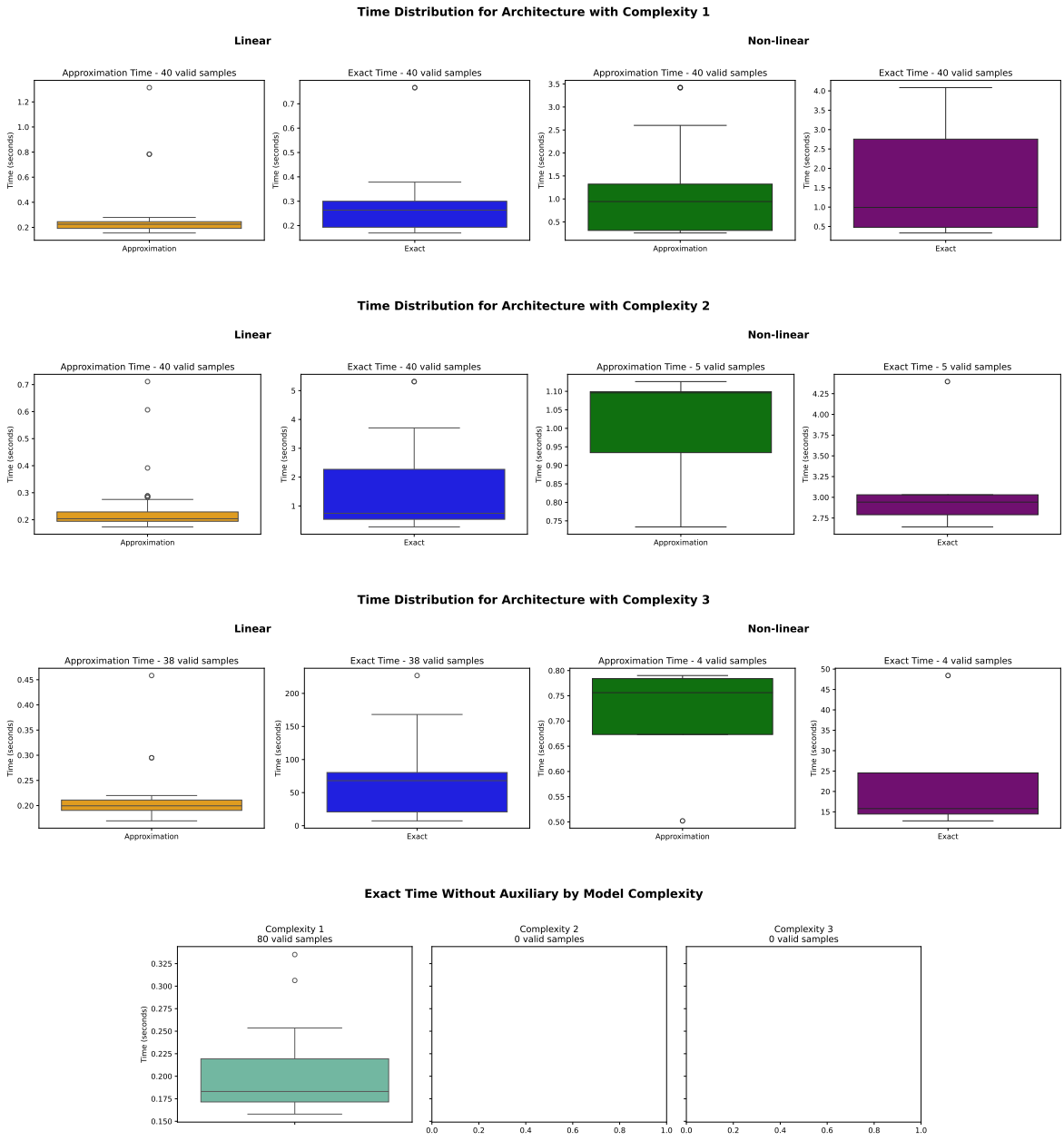
Time Distribution for Architecture with Complexity 1
Linear
Non-linear
Approximation Time - 40 valid samples
Exact Time - 40 valid samples
Approximation Time - 40 valid samples
Exact Time - 40 valid samples
Time (seconds)
Approximation
Exact
Time Distribution for Architecture with Complexity 2
Linear
Non-linear
Approximation Time - 40 valid samples
Exact Time - 40 valid samples
Approximation Time - 5 valid samples
Exact Time - 5 valid samples
Time (seconds)
Approximation
Exact
Time Distribution for Architecture with Complexity 3
Linear
Non-linear
Approximation Time - 38 valid samples
Exact Time - 38 valid samples
Approximation Time - 4 valid samples
Exact Time - 4 valid samples
Time (seconds)
Approximation
Exact
Exact Time Without Auxiliary by Model Complexity
Complexity 1
80 valid samples
Complexity 2
0 valid samples
Complexity 3
0 valid samples
Time (seconds)

Figure 5

Loss	Complexity	# Differs (/20)	# Not Comparable
$\mathbb{L}$	1	16	0
	2	14	0
	3	15	0
$\mathbb{L}_1$	1	6	0
	2	8	0
	3	4	10
$\mathbb{L}_2$	1	11	0
	2	8	0
	3	3	9
$\mathbb{L}_3$	1	11	0
	2	8	0
	3	3	9

Table 4: Comparison between x_{approx} and x_{exact} across different losses and complexities.

However, introducing the regularization term appears to increase the average number of times that x_{approx} and x_{exact} are equal. This is due to a smaller clipping range, which restricts the possible values of the embedding of x, making it more likely for the two verification methods to reach the same result.

4 Conclusion

Overall, the results obtained align well with our expectations. The training phase proceeded as expected, with all configurations exhibiting stable convergence and consistent loss reduction, while reducing the clipping box magnitude.

As anticipated, the approximated verification method proved to be significantly more efficient than the exact method in terms of computation time. Furthermore, the false positive analysis yielded very promising results. With a false positive occurrence as low as 1.67%–3.33%, the approximated method can effectively replace the more computationally expensive exact verification in certain contexts, enabling the safe training of neural networks.

These findings open the door to further investigations. In particular, the approximated verification approach, thanks to its scalability, could be applied to more complex network architectures and real-world scenarios.

A fascinating future case study would be to combine the custom loss functions developed in this work with safe training, and then apply the resulting model to a classification task. This would allow us to observe how accuracy evolves as the clipping box becomes smaller, providing further insights into the trade-off between model safety and performance.

References

[1] M. Francobaldi and M. Lombardi, "Smle: Safe machine learning via embedded overapproximation," 2024.

[2] "Omlt: Optimization and machine learning toolkit." https://omlt.readthedocs.io/en/latest/, 2023. Accessed: 2025-05-26.