Deep Learning Project Report

David Flores and Kirby Linvill

CU Boulder

Abstract. Many state-of-the-art neural networks have been shown to be vulnerable to attacks and adversarial inputs. One solution to this problem is to formally verify that a network is robust against classes of attacks. Unfortunately, the current state-of-the-art neural network verifiers restrict the network architecture to only use the ReLU, sigmoid, and tanh activation functions. We tackle this shortcoming by extending the ETH Robustness Analyzer for Neural Networks (ERAN) to support the ELU activation function. We then demonstrate that the extended ERAN performs comparably on networks using the ELU activation function as on networks using only the previously supported activation functions.

Keywords: Verification, Neural Networks, Robustness, Abstract Interpretation

1 Introduction

Common neural networks have been shown to be vulnerable to adversarial attacks [1,2,3,4,5,6,7,8]. Adding even slight perturbations to an input that are undetectable by humans can cause a trained neural network to output wildly different results. Unfortunately, exhaustively testing for perturbations which give divergent results is impractical, especially given large input spaces when using images. This is where verification methods for neural networks come in, as these techniques allow us to specify constraints and test whether a network meets those constraints (e.g. guarantees that perturbed inputs predict the same output). There are different approaches to verification, but we focus on the incomplete verification of neural networks. The incomplete approach to verification gives up completeness, meaning that some constraints that actually do hold may not be verifiable and result in a false positive when attempting to verify the network. In exchange, these incomplete approaches to verification scale much better to larger networks than complete verification approaches.

While existing techniques for verification of neural networks exist, they have only been developed for the most common activation functions, such as the ReLU, tanh, and sigmoid activation functions; yet, many networks use other activation functions as well. For instance, the LXMERT network [9] uses GELU activations when performing question answering tasks. Without testing on neural networks that use these different functions, it is uncertain whether existing techniques are applicable or scale well to these networks. Thus, it is not known if existing techniques for incomplete verification of neural networks apply to all networks regardless of activation function.

We extend the existing implementation of some of these verification techniques to handle activation functions not yet examined in the verification literature. In particular, we extend the ETH Robustness Analyzer for Neural Networks (ERAN) [10] with an implementation for the ELU activation function. This requires first formulating relaxations for the ELU function before implementing those relaxations in the ERAN framework. We then test the implementation following the approaches in [11] and [12] to ensure that ERAN with our extensions can verify networks that use ELU functions. We further test our implementation to see how precise our robustness guarantees are compared to guarantees for an equivalent network when using ReLU, sigmoid, and tanh activation functions instead. These tests allow us to understand how well these existing techniques for incomplete verification of neural networks apply to new activation functions, and if it is practical to use this technique on realistic networks utilizing ELU.

2 Related Work

Many previous research works provide formal robustness guarantees for neural networks. These works fall into two categories: those that use complete verification approaches and those that use incomplete verification approaches. Both sets of approaches guarantee soundness, that is they prevent false negatives. Complete approaches also guarantee completeness, that is they also prevent false positives. Incomplete approaches give up on completeness in exchange for better scalability.

2.1 Complete Verification of Neural Networks

A common approach to complete verification is to frame network robustness in terms of logical constraints that can then be checked using SMT solvers [13], [14], [15], [16], [17]. Another common approach leverages branch-and-bound techniques and Mixed Integer and Linear Programming (MILP) solvers [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]. Both of these approaches rely on solving NP-complete problems. As a result, they have trouble scaling to neural networks used today.

2.2 Incomplete Verification of Neural Networks

Incomplete verification approaches utilize relaxations and approximations to produce more efficient, albeit less precise, robustness guarantees. Many previous works use abstract interpretation, a common approach in program analysis, to verify robustness in a particular domain [30], [31], [12], [32], [33], [34], [11], [35], [36], [37], [38]. For example, the interval domain is used to abstract out real-valued variables so that each variable is replaced with the bounds on the values it can take (e.g. $\{[a, b] \mid a, b \in \mathbb{R} \cup \{-\infty, \infty\}\}$). Many other works instead use linear approximations to make bounds computations more efficient [39], [40], [41],

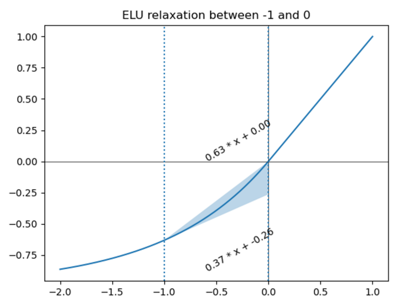
[42], [43], [44], [45], [46], [47], [48]. Others rely on using techniques from simulation [49], duality [50], [51], and semidefinite programming [52], [53], [54]. Another strain of works look instead at probabilistic robustness guarantees rather than deterministic ones using randomized smoothing [55], [56], [57].

Though these works on both complete and incomplete approaches have shown impressive results formally verifying the robustness of neural networks, they only implement and evaluate ReLU, sigmoid, and tanh activation functions. As a result, these works cannot verify common networks including the LXMERT network with a question answering head that uses GELU [58] activations. To move towards supporting verification of these networks, we extend the ETH Robustness Analyzer for Neural Networks (ERAN) [10] to support the common ELU [59] activation functions. ERAN is a state-of-the-art verification tool based on abstract interpretation and the work done in [31], [12], [40], [38], [42], [43], [34], and [11]. To the best of our knowledge, our work is the first time neural networks with these activation functions have been supported by formal robustness analyses.

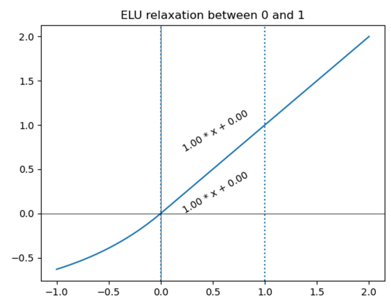
3 Methods

ERAN relies on the ETH Library for Numerical Analysis (ELINA) [60] to implement support for numerical abstract domains. In particular, ELINA includes the code responsible for handling the relaxations for each layer in a neural network including activation layers. As a result, we supply our own relaxations for the ELU activation function as extensions of DeepPoly [12] within ERAN. This requires providing sound upper and lower bounds along with upper and lower polyhedral constraints of the activation functions' outputs as a function of their input values. The polyhedral constraints for variable x_i have the form $v + \sum_j w_j \cdot x_j$ where $v \in \mathbb{R} \cup \{-\infty, \infty\}$, $w \in \mathbb{R}^n$, $\forall j \geq i.\ w_j = 0$. Node outputs are represented by variables x_i that are uniquely numbered such that any node that receives inputs from other nodes must be given a greater number than their inputs. As a result, constraints for variables x_i can refer to earlier variables in the network but not later variables since for all $j \geq i$, $w_j = 0$.

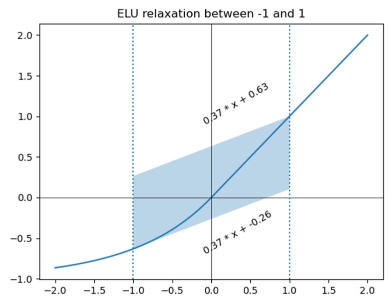
We adapt the existing implementations of the sigmoid and tanh bounds and constraints to implement ELU bounds and constraints. The sigmoid and tanh constraints rely on a generic transformation that requires the activation function $g : \mathbb{R} \to \mathbb{R}$ to be twice differentiable and satisfy $g'(x) > 0$ and $0 \leq g''(x) \Leftrightarrow x \leq 0$. ELU meets this requirement so we can directly adapt the constraints and bounds from sigmoid and tanh. In particular, this means we set the lower and upper bounds for ELU as $l_j = \text{elu}(l_i)$ and $u_j = \text{elu}(u_i)$ where l_i and u_i are the input lower and upper bounds respectively. It also means we set the polyhedral constraints as follows. Let λ be the slope of the line between $\text{elu}(l_i)$ and $\text{elu}(u_i)$. That is, $\lambda = (\text{elu}(u_i) - \text{elu}(l_i))/(u_i - l_i)$. If we are solely in the linear region, that is $l_i \geq 0$, then we set the lower constraint to $\text{elu}(l_i) + \lambda \cdot (x_i - l_i)$. Otherwise, we set the lower constraint to $\text{elu}(l_i) + \min(\text{elu}'(l_i), \text{elu}'(u_i)) \cdot (x_i - l_i)$. If we are solely in the exponential region, that is $u_i \leq 0$, then we set the upper



(a) Our approximation is tight in the exponential region of ELU.



(b) And our approximation is exact in the linear region of ELU.



(c) Yet our approximation is loose when spanning both the linear and exponential regions of ELU.

Fig. 1: ELU Relaxation

constraint to $\mathrm{elu}(u_i) + \lambda \cdot (x_i - u_i)$. Otherwise, we set the upper constraint to $\mathrm{elu}(u_i) + \min(\mathrm{elu}'(l_i), \mathrm{elu}'(u_i)) \cdot (x_i - u_i)$. As shown in Fig. 1, these approximations are tight in the exponential region (1a) and exact in linear region (1b). However, the approximations are loose when the lower and upper bounds span both the linear and exponential regions (1c). A tighter approximation is left for future work.

4 Soundness of ELU Relaxation

To ensure our relaxations are sound, we need to show that they over-approximate the output space. We can show soundness by showing that any output from the activation function is contained between the predicted lower and upper constraints and that the abstract relaxations over-approximate the concrete relaxations. For DeepPoly, the first property is formalized as $\gamma_i(a') \subseteq \bigtimes_{k \in [i]} [l'_k, u'_k]$ where a' is any relaxed output of the activation function (in its abstracted form) and γ is the concretization function for the abstract domain. The second property is formalized as $T_f(\gamma_{i-1}(a)) \subseteq \gamma_i(a')$ where a is the abstract relaxed input of activation function and T_f is a concrete transformer that implements the relaxation.

The general relaxation for sigmoid and tanh is shown to be sound in [12]. We claim our ELU approximation is sound by the same proof since we adapt the same scheme. However, we provide a sketch of soundness in the concrete domain for the reader. We show overall soundness by showing soundness for three cases: when the relaxation is solely in the linear region of ELU ($x \geq 0$), when the relaxation is solely in the exponential region of ELU ($x \leq 0$), and when the relaxation covers both regions.

Across all three cases we let $g_u(x)$ represent the upper constraint and $g_l(x)$ represent the lower constraint for our relaxation. To show soundness, we simply need to show that for all $l_i \leq x \leq u_i$ both $g_u(x) \geq \mathrm{elu}(x)$ and $g_l(x) \leq \mathrm{elu}(x)$.

4.1 Soundness in the Exponential Region

We first consider the upper constraint $g_u(x)$. In the exponential region, $g_u(x) = \mathrm{elu}(u_i) + \lambda \cdot (x - u_i)$ where $\lambda = (\mathrm{elu}(u_i) - \mathrm{elu}(l_i))/(u_i - l_i)$. Consequently $g_u(u_i) = \mathrm{elu}(u_i)$ and $g_u(l_i) = \mathrm{elu}(l_i)$. Since both $\mathrm{elu}'(x) > 0$ and $g_u'(x) > 0$, that is, elu and g_u are both monotonically increasing, we can conclude that $\max(g_u(x)) = g_u(u_i) = \mathrm{elu}(u_i) = \max(\mathrm{elu}(x))$ and $\min(g_u(x)) = g_u(l_i) = \mathrm{elu}(l_i) = \min(\mathrm{elu}(x))$ within the bounds $l_i \leq x \leq u_i$. We can also see that $\mathrm{elu}''(x) > 0$ in this region meaning that the function is convex.

Since elu is convex, we know that $s\,\mathrm{elu}(l_i) + (1-s)\,\mathrm{elu}(u_i) \geq \mathrm{elu}(s \cdot l_i + (1-s) \cdot u_i)$ for all $s \in [0,1]$. Furthermore, since g_u is linear, it follows that $sg_u(l_i) + (1-s)g_u(u_i) = g_u(s \cdot l_i + (1-s) \cdot u_i)$ For all $l_i \leq x \leq u_i$, there exists $s \in [0,1]$ such that $x = s \cdot l_i + (1-s) \cdot u_i$. So, we see that

$$\begin{aligned}
g_u(x) &= g_u(s \cdot l_i + (1-s) \cdot u_i) \\
&= sg_u(l_i) + (1-s)g_u(u_i) \\
&= s\,\mathrm{elu}(l_i) + (1-s)\,\mathrm{elu}(u_i) \\
&\geq \mathrm{elu}(s \cdot l_i + (1-s) \cdot u_i) \\
&= \mathrm{elu}(x).
\end{aligned}$$

Thus we have $g_u(x) \geq \mathrm{elu}(x)$, which meets our definition of soundness for the upper constraint.

Similarly, the lower constraint in the exponential region is $g_l(x) = \mathrm{elu}(l_i) + \min(\mathrm{elu}'(l_i), \mathrm{elu}'(u_i)) \cdot (x - l_i) = \mathrm{elu}(l_i) + \mathrm{elu}'(l_i) \cdot (x - l_i)$. Consequently $g_l(l_i) = \mathrm{elu}(l_i)$. Since $\mathrm{elu}'(x) \geq 0$ and $g_l'(x) \geq 0$, that is, elu and g_l are both monotonically increasing, we can conclude that $\min(g_l(x)) = g_l(l_i)$ and $\mathrm{elu}(l_i) = \min(\mathrm{elu}(x))$ which means that $\min(g_l(x)) = \min(\mathrm{elu}(x))$. Additionally, because $\mathrm{elu}''(x) \geq 0$, such that elu' is monotonically increasing, we know that $\min(\mathrm{elu}'(x)) = \mathrm{elu}'(l_i)$ and that $g_l'(x) \leq \mathrm{elu}'(x)$ when $l_i \leq x$. Since $\min(g_l(x)) = \min(\mathrm{elu}(x))$ and

$g_l'(x) \leq \mathrm{elu}'(x)$, we see that

$$\frac{d}{dx}\left(\mathrm{elu}(x) - g_l(x)\right) = \mathrm{elu}'(x) - g_l'(x) \geq 0,$$

and $\mathrm{elu}(l_i) - g_l(l_i) \geq 0$. Thus, we see that for $l_i \leq x \leq u_i$, $\mathrm{elu}(x) - g_l(x)$ is monotonically increasing, with minimum value $\mathrm{elu}(l_i) - g_l(l_i) = 0$, and hence $\mathrm{elu}(x) - g_l(x) \geq 0$. This can be rearranged to $\mathrm{elu}(x) \geq g_l(x)$, which meets our definition of soundness for the lower constraint.

Our relaxation is therefore sound in the exponential region since both the upper and lower constraints are sound.

4.2 Soundness in the Linear Region

We again first consider the upper constraint $g_u(x)$. In this region, we recall that $g_u(x) = \mathrm{elu}(u_i) + \min(\mathrm{elu}'(l_i), \mathrm{elu}'(u_i)) \cdot (x - u_i)$, which we can simplify to $g_u(x) = u_i + 1 \cdot (x - u_i) = x$ as $\mathrm{elu}(x) = x$ and hence $\mathrm{elu}'(l_i) = \mathrm{elu}'(u_i) = 1$ in the linear region. Hence, we see that $g_u(x) = \mathrm{elu}(x)$. This is both an exact and sound approximation.

Similarly for the lower constraint it is the case that $\lambda = (\mathrm{elu}(u_i) - \mathrm{elu}(l_i))/(u_i - l_i) = (u_i - l_i)/(u_i - l_i) = 1$, and hence $g_l(x) = \mathrm{elu}(l_i) + \lambda \cdot (x - l_i) = l_i + (x - l_i) = x$. So, $g_l(x) = \mathrm{elu}(x)$ and the lower constraint is both an exact and sound approximation.

Our relaxation is therefore both sound and exact in the linear region since both the upper and lower constraints are both sound and exact.

4.3 Soundness When Spanning Both Regions

We again first consider the upper constraint $g_u(x)$. In this case, $g_u(x) = \mathrm{elu}(u_i) + \min(\mathrm{elu}'(l_i), \mathrm{elu}'(u_i)) \cdot (x - u_i) = \mathrm{elu}(u_i) + \mathrm{elu}'(l_i) \cdot (x - u_i)$. Consequently $g_u(u_i) = \mathrm{elu}(u_i)$. Since $\mathrm{elu}'(x) \geq 0$ and $g_u'(x) \geq 0$, such that elu and g_u are monotonically increasing, we conclude that $\max(g_u(x)) = g_u(u_i)$ and $\mathrm{elu}(u_i) = \max(\mathrm{elu}(x))$ which means that $\max(g_u(x)) = \max(\mathrm{elu}(x))$ when $x \leq u_i$. Additionally, because $\mathrm{elu}''(x) \geq 0$, such that elu' is monotonically increasing, we know $\min(\mathrm{elu}'(x)) = \mathrm{elu}'(l_i)$ and hence $\mathrm{elu}'(x) \geq g_u'(x)$ when $x \geq l_i$. Since $\max(g_u(x)) = \max(\mathrm{elu}(x))$ and $\mathrm{elu}'(x) \geq g_u'(x)$, we see that

$$\frac{d}{dx}\left(g_u(x) - \mathrm{elu}(x)\right) = g_u'(x) - \mathrm{elu}'(x) \geq 0,$$

and $g_u(l_i) - \mathrm{elu}(l_i) \geq 0$. Thus, we see that for $l_i \leq x \leq u_i$, $g_u(x) - \mathrm{elu}(x)$ is monotonically increasing, with minimum value $g_u(l_i) - \mathrm{elu}(l_i) = 0$, and hence $g_u(x) - \mathrm{elu}(x) \geq 0$. This can be rearranged to $\mathrm{elu}(x) \leq g_u(x)$, which meets our definition of soundness for the upper constraint.

The proof of soundness for the lower constraint in this case is identical to the proof of soundness for the lower constraint in the exponential region. We can therefore conclude our relaxation is sound when spanning both the linear and exponential regions since both the upper and lower constraints are sound.

5 Experimental Design

To validate and evaluate our ELU relaxation and implementation, we trained and verified models using the ELU activation function against the MNIST dataset and compared the results to models using the ReLU, sigmoid, and tanh activation functions. We used the same model architecture for all models, only varying the activation functions. The models have 5 fully connected hidden layers with 100 neurons per layer. We trained the models in DiffAI [36] using two different approaches: one using ordinary training techniques (the Point domain in DiffAI) and one using a defensive training technique (a mix of the Projected Gradient Descent [61], or PGD, and the BiAdv domain using Iterated Fast Gradient Sign Method [2], or IFGSM). Specifically we used the domain recommended by DiffAI:

```
Mix(a=PGD(r=3,k=16,restart=2, w=0.1),
    b=BiAdv(a=IFGSM(k=5, w=0.05)), bw=0.1)
```

The models were all trained for 10 epochs using the MNIST dataset in DiffAI. All input images were scaled so that pixel values were in the range $[0, 1]$.

Once the models were trained, we tested each model using ERAN on a series of verification trials. The verification challenges we tested included L_∞-norm perturbations [4] of various sizes, with $0.005 \leq \epsilon \leq 0.1$ and centers drawn from ERAN's test set of 100 MNIST images. The L_∞-norm perturbations randomly adjust each pixel value by up to $\pm\epsilon$. Since our pixel values and ϵ values are in the range $[0, 1]$, ϵ can also be seen as a percentage of a maximum possible pixel value. In addition to testing against ERAN's test set, we also tested against the MNIST test dataset in DiffAI. We gathered accuracy metrics for both test sets, as well as the number of correctly predicted images that were robust for perturbations up to ϵ. Accuracy was a reasonable metric given that our test datasets were well balanced.

The code we used to do the training is available on github at `https://github.com/DeepLearningVerificationProject`. All training and evaluation was done using the following commits:

```
DiffAI: c60d5bfdfece42bb66347ed6e5120e634ac9ccd3
ERAN:   3ce9d0adfe789dbef30406c60f6c321d9c7df9ac
ELINA:  e1b1d463b20b4d096e26fb464d0516f99837b7be
```

6 Experimental Results

As shown in Fig. 2, we were able to verify the ELU models for small values of epsilon for all images that were correctly classified. This shows that our extension functionally works. However, there were several notable trends that emerged from our results.

As expected, the number of correct predictions that could be verified to be robust against L_∞-norm perturbations with given ϵ decreased as ϵ increased. ϵ represents the magnitude of the perturbations in the range [0, 1] such that $\epsilon = 0$

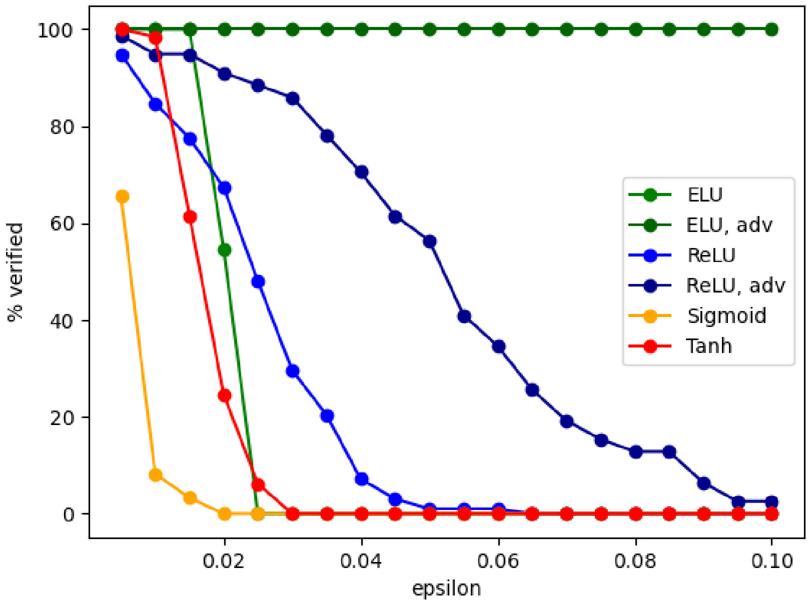


Fig. 2: Percentage of correct predictions verified to be robust. The adv models were trained using defensive techniques to increase model robustness.

corresponds to no perturbation while $\epsilon = 1$ corresponds to perturbations that can push any pixel to any value between its min and max ($[0, 255]$ in the MNIST dataset). Unsurprisingly, very few predictions could be verified to be robust at $\epsilon = 0.1$. It is also unsurprising that networks that were trained without defensive techniques were much less robust than their defensively trained counterparts.

However, it is surprising that the defensively trained ELU network could verify all predictions to be robust up to large values of ϵ, including $\epsilon = 1$. Coupled with the fact that the defensively trained ELU network could only correctly predict $\sim 10\%$ of the ERAN test images as shown in Table 1, these facts imply that the model predicts the same label regardless of the input. Indeed upon further inspection the model always predicted the label 3 when run in ERAN, yet was 98% accurate when manually tested against the images ERAN uses as its test set. When ERAN runs a model against the test set, it does not use the concrete model for predictions but rather the abstract model that utilizes the relaxations for the activation functions. From the set of possible output labels, it then chooses the first label as the prediction. So, if the abstract model is imprecise enough, ERAN will always predict the same label for the model regardless of the input. It is interesting that only the defensively trained ELU model predicts the same label without fail, but this phenomena is likely explained by the fact that our relaxation of ELU is tight in the exponential region and exact in the linear region, so the normally trained ELU model may simply be using ELU more frequently in either the exponential or linear regions.

Table 1: Model performance on the test set ($n = 2000$) and on ERAN's test set ($n = 100$).

activation function	defensively trained	test accuracy	ERAN test accuracy
ELU	no	0.98	0.11
ELU	yes	0.94	0.14
ReLU	no	0.98	0.98
ReLU	yes	0.87	0.78
Sigmoid	no	0.97	0.61
Tanh	no	0.96	0.65

We believe the same root cause is behind the fact that the ReLU models performed comparably on the test set and the ERAN's test set while all the other models performed dramatically worse on ERAN's test set. The ReLU relaxation is exact in both the linear and exponential regions and tight when spanning both regions unlike the sigmoid, tanh, and ELU relaxations. It is likely that the dramatic drop in performance is primarily due to the abstract model being too over-approximate thereby leading to more incorrect predictions. The slight drop in performance for the defensively trained ReLU model could be either an artifact of the small size of ERAN's test set or due to the ReLU activations in the model frequently operating across both the linear and exponential regions (thereby leading to a less precise abstract model). Improved performance on the ERAN test set could likely be achieved by developing tighter approximations of the activation functions.

A natural next step for this work is to conduct a closer examination of the abstract models to validate or disprove our hypothesis about the poor performance on the ERAN test set. Assuming our hypothesis is correct, it would be very valuable to develop tighter approximations of the activation functions. Tighter approximations could be achieved by using a different abstract domain, better relaxations, or by moving to multi-neuron approximations instead of the single-neuron approximations we use in this paper. Further work could also add support for more activation functions including the GELU activation function which is widely used in state-of-the-art transformers.

7 Conclusions

In this work, we investigate if current neural network verification work can be extended to include networks using less common activation functions such as the ELU activation function. To this end, we create and prove sound an approximation of the ELU activation function, and use the approximation to run verification experiments on the networks in question. This work shows some promise in the extension to the ELU activation function, however our experiments also generated some unexpected results. For the images for which the ELU model predicted the correct output, a similar percentage of the corresponding constraints were able to be verified as when using other models with already tested activation functions. Unfortunately, only a small number of images in the verification set were correctly predicted using the ELU model. Nevertheless, we believe that a more precise approximation of the ELU function may fix the issue.

This work helps pushes the boundary of what type of networks are verifiable, which is useful for developers looking to show robustness of their networks. While verifying a network can reduce the attack surface for these networks to be manipulated by bad actors, it does not necessarily provide a guarantee of fairness, or address other potential ethical issues. It is up to the network designers to ensure the constraints they are verifying are sufficient for the use case of the network. For example, a network for use in a self driving car should likely be subject to stronger constraints, such as a larger epsilon value for L_∞-norm perturbations, than a image classifier for a creating captions on sites like YouTube. Similarly, it may be possible to create the proper constraints that can be verified to show some degree of fairness, it is up to the network designers to ensure fairness.

References

1. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
2. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
3. Kurakin, A., Goodfellow, I., Bengio, S., et al.: Adversarial examples in the physical world (2016)
4. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp), IEEE (2017) 39–57
5. Athalye, A., Engstrom, L., Ilyas, A., Kwok, K.: Synthesizing robust adversarial examples. In: International conference on machine learning, PMLR (2018) 284–293
6. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 1625–1634
7. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. (2017) 506–519
8. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016)
9. Tan, H., Bansal, M.: LXMERT: Learning cross-modality encoder representations from transformers. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, Association for Computational Linguistics (November 2019) 5100–5111
10. Müller, M.N., Singh, G., Balunovic, M., Makarchuk, G., Ruoss, A., Serre, F., Baader, M., Cohen, D.D., Gehr, T., Hoffman, A., Maurer, J., Mirman, M., Müller, C., Püschel, M., Tsankov, P., Vechev, M.: ETH Robustness Analyzer for Neural Networks (ERAN)
11. Müller, M.N., Makarchuk, G., Singh, G., Püschel, M., Vechev, M.: Prima: general and precise neural network certification via scalable convex hull approximations. Proceedings of the ACM on Programming Languages **6**(POPL) (2022) 1–33
12. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages **3**(POPL) (2019) 1–30
13. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: International Symposium on Automated Technology for Verification and Analysis, Springer (2017) 269–286
14. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: International conference on computer aided verification, Springer (2017) 3–29
15. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: International conference on computer aided verification, Springer (2017) 97–117
16. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification, Springer (2019) 443–452

17. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: International Conference on Computer Aided Verification, Springer (2010) 243–257
18. Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming **183**(1) (2020) 3–39
19. Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., Misener, R.: Efficient verification of relu-based neural networks via dependency analysis. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 34. (2020) 3291–3299
20. Bunel, R., Mudigonda, P., Turkaslan, I., Torr, P., Lu, J., Kohli, P.: Branch and bound for piecewise linear neural network verification. Journal of Machine Learning Research **21**(2020) (2020)
21. Lu, J., Kumar, M.P.: Neural network branching for neural network verification. arXiv preprint arXiv:1912.01329 (2019)
22. De Palma, A., Behl, H.S., Bunel, R., Torr, P., Kumar, M.P.: Scaling the convex barrier with active sets. In: Proceedings of the ICLR 2021 Conference, Open Review (2021)
23. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356 (2017)
24. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, J.Z.: Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. arXiv preprint arXiv:2103.06624 (2021)
25. Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., Hsieh, C.J.: Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. arXiv preprint arXiv:2011.13824 (2020)
26. Cheng, C.H., Nührenberg, G., Ruess, H.: Maximum resilience of artificial neural networks. In: International Symposium on Automated Technology for Verification and Analysis, Springer (2017) 251–268
27. Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**(3) (2018) 296–309
28. Bunel, R.R., Turkaslan, I., Torr, P., Kohli, P., Mudigonda, P.K.: A unified view of piecewise linear neural network verification. Advances in Neural Information Processing Systems **31** (2018)
29. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks. In: NASA Formal Methods Symposium, Springer (2018) 121–138
30. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP), IEEE (2018) 3–18
31. Singh, G., Gehr, T., Mirman, M., Püschel, M., Vechev, M.: Fast and effective robustness certification. Advances in neural information processing systems **31** (2018)
32. Li, J., Liu, J., Yang, P., Chen, L., Huang, X., Zhang, L.: Analyzing deep neural networks with symbolic propagation: Towards higher precision and faster verification. In: International Static Analysis Symposium, Springer (2019) 296–319
33. Urban, C., Christakis, M., Wüstholz, V., Zhang, F.: Perfectly parallel fairness certification of neural networks. Proceedings of the ACM on Programming Languages **4**(OOPSLA) (2020) 1–30

34. Müller, C., Serre, F., Singh, G., Püschel, M., Vechev, M.: Scaling polyhedral neural network verification on gpus. Proceedings of Machine Learning and Systems **3** (2021) 733–746
35. Zhang, H., Shinn, M., Gupta, A., Gurfinkel, A., Le, N., Narodytska, N.: Verification of recurrent neural networks for cognitive tasks via reachability analysis. In: ECAI 2020. IOS Press (2020) 1690–1697
36. Mirman, M., Gehr, T., Vechev, M.: Differentiable abstract interpretation for provably robust neural networks. In: International Conference on Machine Learning, PMLR (2018) 3578–3586
37. Müller, C., Singh, G., Püschel, M., Vechev, M.T.: Neural network robustness verification on gpus. CoRR, abs/2007.10868 (2020)
38. Singh, G., Ganvir, R., Püschel, M., Vechev, M.: Beyond the single neuron convex barrier for neural network certification. Advances in Neural Information Processing Systems **32** (2019)
39. Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.J., Daniel, L., Boning, D., Dhillon, I.: Towards fast computation of certified robustness for relu networks. In: International Conference on Machine Learning, PMLR (2018) 5276–5285
40. Singh, G., Gehr, T., Püschel, M., Vechev, M.: Boosting robustness certification of neural networks. In: International conference on learning representations. (2018)
41. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. Advances in neural information processing systems **31** (2018)
42. Balunovic, M., Baader, M., Singh, G., Gehr, T., Vechev, M.: Certifying geometric robustness of neural networks. Advances in Neural Information Processing Systems **32** (2019)
43. Ruoss, A., Baader, M., Balunović, M., Vechev, M.: Efficient certification of spatial robustness. arXiv preprint arXiv:2009.09318 (2020)
44. Ko, C.Y., Lyu, Z., Weng, L., Daniel, L., Wong, N., Lin, D.: Popqorn: Quantifying robustness of recurrent neural networks. In: International Conference on Machine Learning, PMLR (2019) 3468–3477
45. Gowal, S., Dvijotham, K.D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., Kohli, P.: Scalable verified training for provably robust image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 4842–4851
46. Bunel, R.R., Hinder, O., Bhojanapalli, S., Dvijotham, K.: An efficient nonconvex reformulation of stagewise convex optimization problems. Advances in Neural Information Processing Systems **33** (2020) 8247–8258
47. Lyu, Z., Ko, C.Y., Kong, Z., Wong, N., Lin, D., Daniel, L.: Fastened crown: Tightened neural network robustness certificates. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 34. (2020) 5037–5044
48. Tjandraatmadja, C., Anderson, R., Huchette, J., Ma, W., PATEL, K.K., Vielma, J.P.: The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. Advances in Neural Information Processing Systems **33** (2020) 21675–21686
49. Xiang, W., Tran, H.D., Johnson, T.T.: Output reachable set estimation and verification for multilayer neural networks. IEEE transactions on neural networks and learning systems **29**(11) (2018) 5777–5783
50. Wong, E., Kolter, Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: International Conference on Machine Learning, PMLR (2018) 5286–5295

51. Dvijotham, K., Stanforth, R., Gowal, S., Mann, T.A., Kohli, P.: A dual approach to scalable verification of deep networks. In: UAI. Volume 1. (2018) 3
52. Raghunathan, A., Steinhardt, J., Liang, P.: Certified defenses against adversarial examples. arXiv preprint arXiv:1801.09344 (2018)
53. Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R.R., Shankar, S., Steinhardt, J., Goodfellow, I., Liang, P.S., et al.: Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. Advances in Neural Information Processing Systems **33** (2020) 5318–5331
54. Raghunathan, A., Steinhardt, J., Liang, P.S.: Semidefinite relaxations for certifying robustness to adversarial examples. Advances in Neural Information Processing Systems **31** (2018)
55. Cohen, J., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: International Conference on Machine Learning, PMLR (2019) 1310–1320
56. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., Jana, S.: Certified robustness to adversarial examples with differential privacy. In: 2019 IEEE Symposium on Security and Privacy (SP), IEEE (2019) 656–672
57. Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., Yang, G.: Provably robust deep learning via adversarially trained smoothed classifiers. Advances in Neural Information Processing Systems **32** (2019)
58. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
59. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
60. Singh, G., He, J., Müller, C., Serre, F., Ruoss, A., Makarchuk, G., Püschel, M., Vechev, M.: ETH LIbrary for Numerical Analysis (ELINA)
61. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)