Empirical Analysis of Benchmark Generation for the Verification of Neural Network Image Classifiers

Diego Manzanas Lopez[1] and Taylor T. Johnson[1]

Vanderbilt University, Nashville, TN
{diego.manzanas.lopez,taylor.johnson}@vanderbilt.edu

Abstract. Deep Learning success in a wide range of applications, such as image recognition and natural language processing, has led to the increasing usage of this technology in many domains, including safety-critical applications such as autonomous cars and medicine. The usage of the models, e.g., neural networks, in safety critical applications demands a thorough evaluation from a component and system level perspective. In these domains, formal methods have the ability to guarantee the correct operation of these components. Despite great efforts in the formal verification of neural networks in the past decade, several challenges remain. One of these challenges is the development of neural networks for easier verification. In this work, we present an empirical analysis, presented as a Latin Hypercube experiment design, in which we evaluate how regularization and initialization methods across different random seeds on two datasets affect the verification analysis of a reachability analysis technique for the verification of neural networks. We show that there are certain training routines that simplify the formal verification task. Lastly, a discussion on how these training approaches impact the robustness verification and reachability computation of the method utilized is included.

Keywords: Formal Verification, Medical Imaging, Deep Learning, Reachability Analysis

1 Introduction

Neural Networks (NN) are a type of machine learning models that are able to learn complex patterns from data, and have been used to achieve state-of-the-art results in a wide variety of tasks such as image recognition [23,6,14] and natural language processing [7,32,29]. However, their usage in safety-critical domains requires an extensive and rigorous analysis of these models from both a component and system level perspective. Formal methods are techniques that are able to provide guarantees on the functionality of these models to ensure the correct behavior in these domains. In the past several years, there have been numerous formal verification methods and tools developed to address this challenge [41,2,31,28,22,49,27,30,35,48]. Despite recent efforts, several challenges in existing state-of-the-art methods and tools remain due to the complexity of these models and constant and rapid development of new NN architectures and models. One of the main challenges is the scalability of verification methods to large models, such as those use in Semantic Segmentation [31,45] or LLMs [32]. Another challenge is the disconnect between NN development and verification. Typically, first goes the development and training of the neural network, followed by the verification approach on the learned network (fixed parameters). If the networks do not meet the formal requirements or the methods are not able to prove them due to the complexity of the models, either new methods are needed to be developed or a new model is needed to be trained.

In this manuscript, we focus on the latter challenge, with the goal of providing some guidance and understanding on how training procedures affect formal verification methods. This idea of this project began when we discovered a large difference in the reachability computation times (order of 10 to 100 times faster) when analyzing the robustness of a neural network classifier on a medical image dataset. These findings were present not only in individual instances of a class, but were general to the whole set of images analyzed from the same class. After these observations, we decided to dig a little deeper to understand the reason behind these differences. Is it specific to the images evaluated? Is it just for a specific model? What if we change the training method, will a similar verification patter hold for the new method as well?

In this manuscript, we present a set of experiments that provide some insights to these questions using the verification tool NNV [46,30]. The contributions of this work are:

- Introduction of a new benchmark for neural network verification in the area of medical imaging (MedNIST [1]).
- Compute the formal verification of two benchmarks, both trained on gray-scale image datasets, consisting of a total of 45 models and 300 instances analyzed per benchmark.
- Analysis of NNV [41,30] methods on these two benchmarks, including a discussion of the training effects on the reachability method used.

2 Related Work

Neural network verification. The area of neural network verification has grown immensely in recent years, having the community establish and develop standard input formats[1]. These have been especially useful for friendly competitions [28,31] as well as for method and tool comparison that enable a faster and (hopefully) fairer comparison across tools [49,48,21,22,2,30,46,31]. Despite recent efforts, the majority of these methods focus on verifying feedforward and convolutional NN architectures. These approaches can generally be classified into sound or unsound, and complete or incomplete. Unsound approaches are less common for NN verification than sound approaches, as they cannot provide formal guarantees on the computer results. They usually refer to probabilistic analysis such as [42]

[1] vnnlib: `https://www.vnnlib.org`

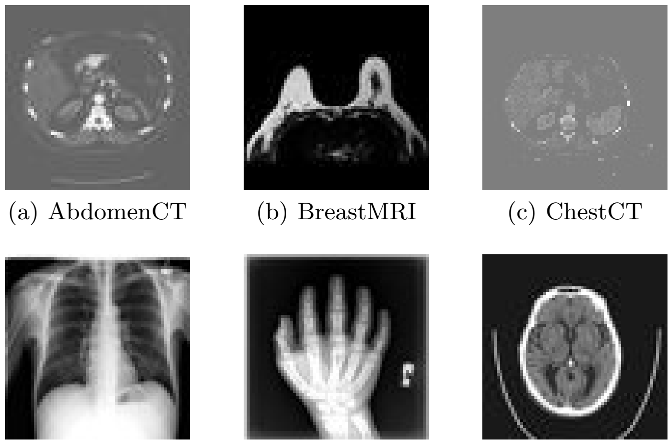
or under approximations of the actual verification result, which are typically faster to compute than sound approximations [17]. Complete and sound methods refer to algorithms that can precisely analyze whether a given property holds on a model, also referred to as exact methods. A disadvantage from these methods is how computationally expensive these are, often becoming prohibitive to compute, thus suffering from scalability issues for large models or inputs sets [45,31]. These methods are often limited in the type of layers and architectures they can be used for. These can be Satisfiability Modulo Theories (SMT) based methods [22,21], Mixed Integer Linear Program (MILP) based methods [40], reachability analysis methods [45], and others such as branch and bound methods [4]. To overcome some of these challenges, sound and incomplete methods have been developed. These methods often introduced a tradeoff between precision, scalability and computational power needed. These methods are capable of computing the verification results faster than sound and complete methods, however, due to the overapproximation computations, unknown results may arise (cannot guarantee specification is violated nor satisfied). Several of these methods are based on abstract interpretation, some of which have demonstrated to outperform complete methods by orders of magnitude (time wise) [31]. Recent work in [11] has enhanced the abstraction-based verification of neural networks via residual reasoning.

Training & Verification & Repair. There have also been some works focusing on repairing or retraining neural networks when a specification is violated [8,39,13]. Many of these efforts focus on fine-tuning or retraining the network when inputs violating the output constraints are found [51,10,33,36], directly modifying the parameters of the network to correct violating inputs [9,16,47], or on modifying the architecture of the neural network to facilitate the repair of the model such as in [37]. Another area that has seen some efforts is to directly train neural networks for enhancing the verification approaches. Some works focus on replacing the ReLU layers by Parametric ReLUs to enhance both robustness and verification scalability [26], others have focused on using stability training methods for ReLUs to pre-estimate the bounds for all ReLU neurons [50], providing local Lipschitz bounds to the networks to simplify its verification [20], using interval bound propagation during training to improve the verified robustness of the models [34], or developing regularization methods for improving robustness and reducing the verification time for autoencoders [5].

3 Evaluation

We use two datasets in our evaluation: MNIST [25], and a medical image dataset, MedNIST [1]. The MedNIST dataset was assembled by B.J. Erickson (Department of Radiology, Mayo Clinic). [2] The dataset contains 58954 medical images belonging to 6 classes: AbdomenCT, BreastMRI, ChestCT, CSR, Hand, and HeadCT, which are depicted in Figure 1. Each of the classes contains 10,000 images except for BreastMRI, which contains 8,954 images.

[2] Available at `https://github.com/Project-MONAI/MONAI/`


(a) AbdomenCT (b) BreastMRI (c) ChestCT

(d) CXR (e) Hand (f) HeadCT

Fig. 1. MedNIST dataset visualization.

We present our study as a Latin Hypercube experiment design. A Latin hypercube typically consists of N variables divided into M equally sized intervals or discrete values, where each sample is unique: each sample is the only one in each axis-aligned hyperplane containing it [12]. Our experimental design consists of 3 variables (*init_method*, *reg_method*, *random_seed*), the first two containing 3 different values and the latter one with 5 possible values. The experiments consist of training 45 different models, 5 models per combination of hyperparameters: initialization *init_method* $\in$ {He [18], Glorot [15], narrow_normal[3]}, and regularization scheme *reg_method* $\in$ {Dropout [38], Jacobian [19], L_2 [24] }. Each of these models are initialized with a different random seed, *random_seed* $\in \{0, 1, 2, 3, 4\}$, to evaluate the training combination of *init_method* $\times$ *reg_method*.

Once all models are trained, we perform a verification analysis, for which we select 300 images from the test dataset, and apply an L_∞ attack with an $\epsilon = \{3/255\}$, for a total of 300 images evaluated for each of the 45 neural networks [4]. To verify the robustness of these networks against adversarial attacks, we only chose images that are correctly classified by all networks, and select the same number of images per class in each dataset: 50 images per class in MedNIST, and 30 for MNIST. Formally, we evaluate the robustness of a neural network $\mathcal{F}(z)$, with input image $z \in \mathbb{R}^{i \times j}$, perturbation parameter $\epsilon \in \mathbb{R}$ and an input set Z_p containing z_p such that $Z_p = \{z : ||z - z_p|| \leq \epsilon\}$ that represents the set of all possible perturbations of z. The neural network is locally **robust** at z if it

[3] Weights are independently sampled from a normal distribution with 0 mean and standard deviation of 0.01

[4] The model architecture is depicted in Figure A in the Appendix.

correctly classifies all the perturbed inputs z_p to the same label as z, i.e., the system is **robust** if $\mathcal{F}(z_p) = \mathcal{F}(z)$ for all $z_p \in Z_p$.

Our goal is to gain some insights to the following questions:

– Is there a training procedure that facilitates the verification of these models?
– Can the training and verification trends hold across datasets?
– How much of an effect has the initial random seed on our evaluation?
– Are there specific classes that are harder or easier to verify than others?

4 Results

We analyze the verification results based on 1) initialization, 2) regularization, and 3) random seed, with respect to every class in the dataset. It is important to remember that the robustness percentage may be greater than the accuracy, as we only evaluated the models in images correctly classified by all of them. On figures 2 to 7, on the left (subfig. *a*) we present the average percentage of instances verified to be robust with respect to each image class, and on the right (subfig. *b*), we present the average computation time to verify each instance with respect to each image class[5]. In addition, we present a summary of the verification results in the Appendix in Tables 1 and 2 for MedNIST and MNIST respectively.

4.1 Initialization

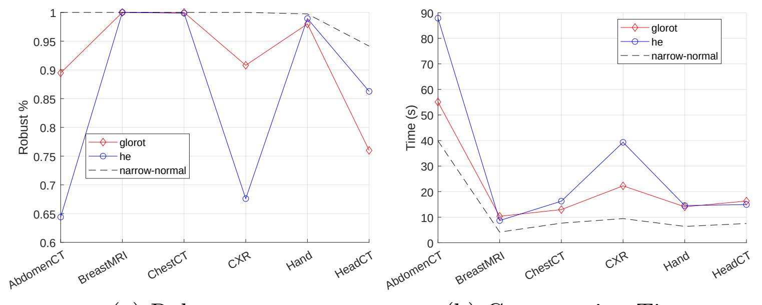
In Fig. 2 we present the average robustness percentage and computation time across all 15 models trained using each initialization method on the MedNIST dataset. We observe that we are able to verify the largest number of instances of the models trained using the *narrow-normal* initialization method, and much faster than the other ones.

In Fig. 3 we present the average robustness percentage and computation time across all 15 models trained using each initialization method on the MNIST dataset. Similar to the MedNIST results but less pronounced, we observe that we are able to verify the largest number of instances of the models trained using the *narrow-normal* initialization method, and faster than the other ones, but very close to the *he* initializer.

4.2 Regularization

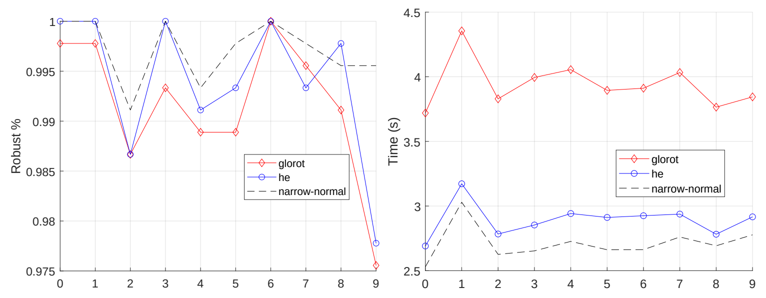
In Fig. 4 we present the average robustness percentage and computation time across all 15 models trained using each regularization method on the MedNIST dataset. We observe that we are able to verify the largest number of instances of the models trained using the *Jacobian* regularization method, and slightly faster than the other ones.

[5] Code is available at: `https://github.com/verivital/nnv/tree/master/code/nnv/examples/Submission/AISOLA2023`



(a) Robustness (b) Computation Time

Fig. 2. *MedNIST Results*. Comparison across initialization schemes with respect to each image class.



(a) Robustness (b) Computation Time

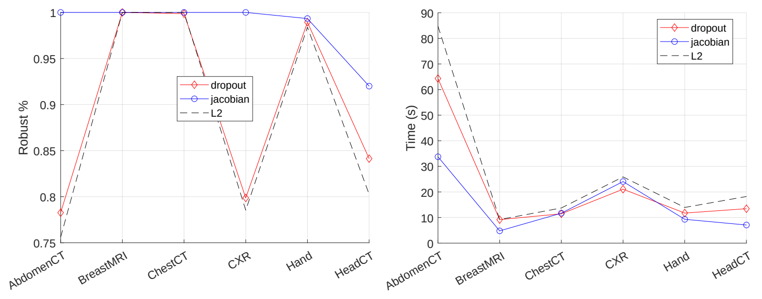
Fig. 3. *MNIST Results*. Comparison across initialization schemes with respect to each image class.

In Fig. 5 we present the average robustness percentage and computation time across all 15 models trained using each regularization method on the MNIST dataset. We observe that we are able to compute the verification result of the models using the dropout regularization method the fastest, about 2× faster than *Jacobian* (fastest on MedNIST), but the number of instances across them is very similar.

4.3 Random seed

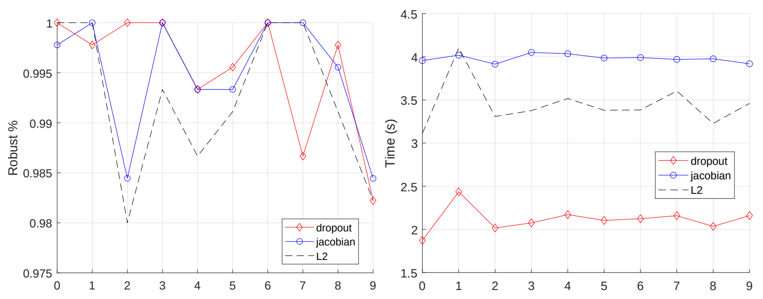
In Fig. 6 we present the average robustness percentage and computation time across all 9 models initialized using each random seed on the MedNIST dataset. Models with random seed *1* are slightly faster to verify, with a larger number of instances verified.

In Fig. 7 we present the average robustness percentage and computation time across all 9 models initialized using each random seed on the MNIST dataset. In



(a) Robustness (b) Computation Time

Fig. 4. *MedNIST Results.* Comparison across regularization techniques with respect to each image class.

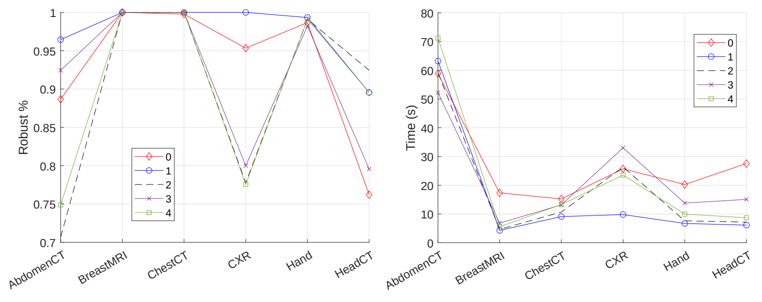


(a) Robustness (b) Computation Time

Fig. 5. *MNIST Results.* Comparison across regularization techniques with respect to each image class.

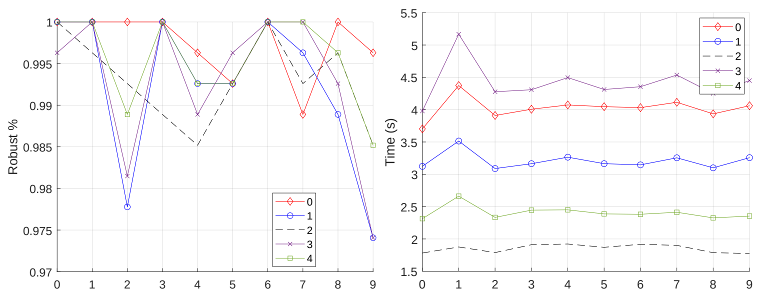
this case, there is not a clear "winner" in terms of number of instances verified, but interestingly, there is a very defined pattern in the verification computation across the random seeds, being *2* the fastest and *3* the slowest.

Regularization & Initialization combinations. Looking into these results, we would expect to have several models with 100 % verified instances on the MNIST dataset (total of 10, as observed in Table 2, and having the fastest models to verify to be a model trained using the narrow-normal initialization with the dropout regularizer, depicted in Fig. 8(b). For the MedNIST dataset, the fastest combination is expected to be a narrow-normal initialized model with either dropout or Jacobian regularization, as observed in Fig. 8(a). A surprising result is that there are 6 models with 100 % verified instances on the MedNIST benchmark, 4 of which are models trained using Jacobian regularization and narrow-normal initialization method, as observed in Table 2.



(a) Robustness (b) Computation Time

Fig. 6. *MedNIST Results*. Comparison across random seeds with respect to each image class.



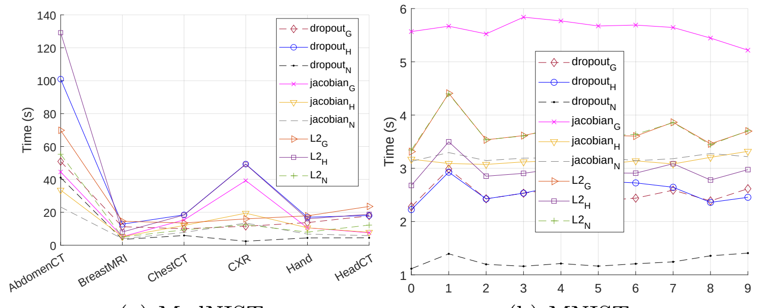
(a) Robustness (b) Computation Time

Fig. 7. *MNIST Results*. Comparison across random seeds with respect to each image class.

5 Discussion

Based on the results from Figures 2 to 7, one can observe that there are trends that hold across the datasets, while some other results are very different from each other. We begin with the similarities between the two datasets in terms of verification results. Looking into the initialization schemes (Figs. 2 and 3), we observe similar results, being *narrow-normal* initializer the fastest to verify across both datasets, and the one with the highest number of instances verified. Another common result from both datasets is L_2 regularizer having the least number of instances verified, and the slowest for MedNIST and second slowest for MNIST, closer to the slowest (Jacobian) than the fastest (dropout).

There are also some clear differences across the datasets. For the MNIST models, the computation time trends are held across all the classes, indicating that the verification computation depends more on the model than the image class evaluated, except for class *1* which is slightly slower. On the other hand,



(a) MedNIST (b) MNIST

Fig. 8. *Combination Results.* Comparison across regularization & initialization combinations in terms of verification computation time in seconds for MedNIST and MNIST benchmarks.

for the MedNIST models, the verification times are very dependent on the image type, as we can observe in Figs 2, 4, and 6, where the verification of *AbdomenCT* images is the slowest, up to 10× slower than other classes, and *CXR* the second slowest with 4× to 5× slower than other classes. *AbdomenCT* is also the class with the least number of instances verified, followed by *HeadCT* and *CXR*.

These last results prompt two questions:

1. Why is there such a big difference in the computation time of the same model when looking at two different images from the same dataset?
2. Do the lower robustness percentages mean the models are less robust, or are these models harder to verify?

To answer these, we need to understand the verification method used. In this paper, we use a sound and incomplete reachability method described in [41,44]. This method represents the sets using ImageStars [41], and computes the output set using a layer-by-layer approach. We run several examples and timed every operation within the reachability computation to understand where the timing difference lies, and discovered (as expected) that the largest percentage of the computation time is in the reachability computation of the ReLU layers. More specifically, in the computation of the solution of the Linear Programming (LP) problems. When using the approach in [44], if the estimated range contains the zero point, we solve two LP problems to get the range of this specific input. The solution to each of these LP problems is an overapproximation of the exact range (sound and incomplete). Thus, it is not a coincidence that the class with the slowest verification computation (AbdomenCT) is also the class with the least number of instances verified. These two variables are correlated, as the larger number of LP problems solved leads to a larger overapproximation of the reachable set of the neural network.

The answer to the latter question is partially covered with the previous one. These models are not necessarily less robust to these image types under L_∞

attacks, but harder to verify due to the accumulated overapproximations in the reachability computation of ReLU layers. In addition, we attempted to find counterexamples to the unknown instances using random examples within the input sets, including the upper and lower bounds of the input set. However, we were only able to find 2 and 1 counterexamples across all unknown instances for the MedNIST and MNIST benchmarks respectively, as depicted in Tables 1 and 2.

Limitations

Dataset Coverage. For the robustness analysis, we randomly select 20 images per class, which is less than 1% of the images in the dataset.

Model architecture. The analysis is limited to a single architecture with a convolutional, a batch normalization, a ReLU, an average pooling and a fully-connected layer. When looking closely at the results, we observe that the *harder* examples (in terms of computation time) tend to *activate* both sides of the ReLU neurons (input interval is less than 0 and greater than 0), requiring to solve a larger amount of Linear Programming (LP) problems, e.g., taking up to 80% to 90% of the total reachability computation time for some instances. Using other activation function or architectures may reduce the complexity of these *harder* instances.

Methods evaluated. We were able to discover some trends on finding harder verification examples for NNV, which is a reachability based tool using Star sets [3,44,46,41]. Although we expect similar results in terms of computation time trends, more evaluations are needed to determine if these trends also hold when using other methods such as SMT or MILP based (e.g., Marabou [22]).

Complete vs incomplete verification. We evaluate the robustness of the networks using a sound but incomplete method, so we can only guarantee the number of instances verified to be robust, but no guarantees on the others. To prove the network is not robust, we would have to find counterexamples, or we would need to run a complete verifier (*exact* reachability methods in NNV) to determine the certified robustness score of each network. However, the goal of this paper is to understand the complexity of the verification analysis on different training routines, and we would expect to observe a similar trend for the exact analysis on the computation time aspect (the larger computation times come from having to compute the reachability of ReLU neurons when the input interval value is less than 0 and greater than 0, which has a similar effect on the complete methods [43]).

6 Conclusion

In this study, we have presented two neural network verification benchmarks, one from the MedNIST dataset [1] and one from MNIST [25] consisting of 45 neural networks and 300 verification instances per network, for each of the benchmarks. On these, we have analyzed how different training procedures affect a sound and incomplete reachability analysis technique implemented in NNV [41,44]. We can observe that there are training combinations that lead to models that are easier to verify using these methods. For the more challenging verification instances, we discuss the reasons behind it: input ranges to the neurons in the ReLU layers are "activating" both sides of the function ($max(0, input)$), requiring to solve a larger number of LP problems, which in turn leads to a larger overapproximation of the output reachable set, making the computation slower and more complex.

Acknowledgements The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 2028001, 2220426 and 2220401, the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-23-C-0518, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019 and FA9550-23-1-0135. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF.

References

1. apolanco3225: Medical mnist classification. `https://github.com/apolanco3225/Medical-MNIST-Classification` (2017)
2. Bak, S.: nnenum: Verification of relu neural networks with optimized abstraction refinement. In: NASA Formal Methods Symposium. pp. 19–36. Springer (2021)
3. Bak, S., Duggirala, P.S.: Simulation-equivalent reachability of large linear systems with inputs. In: Majumdar, R., Kunčak, V. (eds.) Computer Aided Verification. pp. 401–420. Springer International Publishing, Cham (2017)
4. Bunel, R., Turkaslan, I., Torr, P.H.S., Kumar, M.P., Lu, J., Kohli, P.: Branch and bound for piecewise linear neural network verification. J. Mach. Learn. Res. **21**(1) (jan 2020)
5. Böing, B., Müller, E.: On training and verifying robust autoencoders. In: 2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10 (2022). `https://doi.org/10.1109/DSAA54385.2022.10032334`
6. Cireşan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745 (2012)
7. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning. pp. 160–167. ACM (2008)
8. Cruz, U.S., Ferlez, J., Shoukry, Y.: Safe-by-repair: A convex optimization approach for repairing unsafe two-level lattice neural network controllers. In: 2022 IEEE 61st Conference on Decision and Control (CDC). pp. 3383–3388 (2022). `https://doi.org/10.1109/CDC51059.2022.9993239`
9. Dong, G., Sun, J., Wang, J., Wang, X., Dai, T.: Towards repairing neural networks correctly (2021)
10. Dreossi, T., Ghosh, S., Yue, X., Keutzer, K., Sangiovanni-Vincentelli, A., Seshia, S.A.: Counterexample-guided data augmentation. In: Proceedings of the 27th

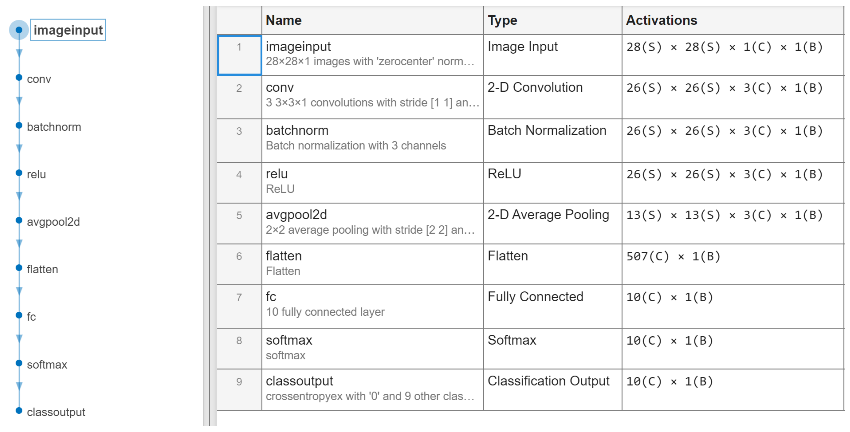
International Joint Conference on Artificial Intelligence. p. 2071–2078. IJCAI'18, AAAI Press (2018)

11. Elboher, Y.Y., Cohen, E., Katz, G.: Neural network verification using residual reasoning. In: Software Engineering and Formal Methods: 20th International Conference, SEFM 2022, Berlin, Germany, September 26–30, 2022, Proceedings. p. 173–189. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-17108-6_11
12. Fisher, R.A.: Statistical methods for research workers. In: Breakthroughs in statistics: Methodology and distribution, pp. 66–70. Springer (1970)
13. Fu, F., Li, W.: Sound and complete neural network repair with minimality and locality guarantees. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=xS8AMYiEav3
14. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2414–2423 (2016)
15. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010)
16. Goldberger, B., Katz, G., Adi, Y., Keshet, J.: Minimal modifications of deep neural networks using verification. In: Albert, E., Kovacs, L. (eds.) LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 73, pp. 260–278. EasyChair (2020). https://doi.org/10.29007/699q, https://easychair.org/publications/paper/CWhF
17. Goubault, E., Putot, S.: Rino: Robust inner and outer approximated reachability of neural networks controlled systems. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification. pp. 511–523. Springer International Publishing, Cham (2022)
18. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 1026–1034 (2015). https://doi.org/10.1109/ICCV.2015.123
19. Hoffman, J., Roberts, D.A., Yaida, S.: Robust learning with jacobian regularization (2019)
20. Huang, Y., Zhang, H., Shi, Y., Kolter, J.Z., Anandkumar, A.: Training certifiably robust neural networks with efficient local lipschitz bounds. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 22745–22757. Curran Associates, Inc. (2021)
21. 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. pp. 97–117. Springer (2017)
22. 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. pp. 443–452. Springer (2019)
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

24. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Proceedings of the 4th International Conference on Neural Information Processing Systems. p. 950–957. NIPS'91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1991)
25. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist **2** (2010)
26. Leofante, F., Henriksen, P., Lomuscio, A.: Verification-friendly networks: the case for parametric relus. In: Workshop on Formal Verification of Machine Learning, Colocated with ICML 2022. IEEE (2022)
27. Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M.J.: Algorithms for verifying deep neural networks. Foundations and Trends in Optimization **4**(3-4), 244–404 (2021). https://doi.org/10.1561/2400000035
28. Lopez, D.M., Althoff, M., Benet, L., Chen, X., Fan, J., Forets, M., Huang, C., Johnson, T.T., Ladner, T., Li, W., Schilling, C., Zhu, Q.: Arch-comp22 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In: Frehse, G., Althoff, M., Schoitsch, E., Guiochet, J. (eds.) Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22). EPiC Series in Computing, vol. 90, pp. 142–184. EasyChair (2022)
29. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
30. Manzanas Lopez, D., Choi, S.W., Tran, H.D., Johnson, T.T.: Nnv 2.0: The neural network verification tool. In: Enea, C., Lal, A. (eds.) Computer Aided Verification. pp. 397–412. Springer Nature Switzerland, Cham (2023)
31. Müller, M.N., Brix, C., Bak, S., Liu, C., Johnson, T.T.: The third international verification of neural networks competition (vnn-comp 2022): Summary and results (2022)
32. OpenAI: Gpt-4 technical report (2023)
33. Ren, X., Yu, B., Qi, H., Juefei-Xu, F., Li, Z., Xue, W., Ma, L., Zhao, J.: Few-shot guided mix for dnn repairing. In: Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020. pp. 717–721. Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020, Institute of Electrical and Electronics Engineers Inc., United States (Sep 2020). https://doi.org/10.1109/ICSME46990.2020.00079
34. Shi, Z., Wang, Y., Zhang, H., Yi, J., Hsieh, C.J.: Fast certified robust training with short warmup. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 18335–18349. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper_files/paper/2021/file/988f9153ac4fd966ea302dd9ab9bae15-Paper.pdf
35. Singh, G., Gehr, T., Mirman, M., Püschel, M., Vechev, M.: Fast and effective robustness certification. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018)
36. Sinitsin, A., Plokhotnyuk, V., Pyrkin, D., Popov, S., Babenko, A.: Editable neural networks. In: International Conference on Learning Representations (2020)
37. Sotoudeh, M., Thakur, A.V.: Provable repair of deep neural networks. In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. p. 588–603. PLDI 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3453483.3454064

38. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014), http://jmlr.org/papers/v15/srivastava14a.html
39. Tao, Z., Nawas, S., Mitchell, J., Thakur, A.V.: Architecture-preserving provable repair of deep neural networks. Proc. ACM Program. Lang. **7**(PLDI) (jun 2023). https://doi.org/10.1145/3591238
40. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356 (2017)
41. Tran, H.D., Bak, S., Xiang, W., Johnson, T.T.: Verification of deep convolutional neural networks using imagestars. In: 32nd International Conference on Computer-Aided Verification (CAV). Springer (July 2020)
42. Tran, H.D., Choi, S., Okamoto, H., Hoxha, B., Fainekos, G., Prokhorov, D.: Quantitative verification for neural networks using probstars. In: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control. HSCC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3575870.3587112
43. Tran, H.D., Musau, P., Lopez, D.M., Yang, X., Nguyen, L.V., Xiang, W., Johnson, T.T.: Star-based reachability analsysis for deep neural networks. In: 23rd International Symposisum on Formal Methods (FM'19). Springer International Publishing (October 2019)
44. Tran, H.D., Pal, N., Lopez, D.M., Musau, P., Yang, X., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: Verification of piecewise deep neural networks: A star set approach with zonotope pre-filter. Form. Asp. Comput. **33**(4–5), 519–545 (aug 2021)
45. Tran, H.D., Pal, N., Musau, P., Yang, X., Hamilton, N.P., Lopez, D.M., Bak, S., Johnson, T.T.: Robustness verification of semantic segmentation neural networks using relaxed reachability. In: 33rd International Conference on Computer-Aided Verification (CAV). Springer (July 2021)
46. Tran, H.D., Yang, X., Lopez, D.M., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: 32nd International Conference on Computer-Aided Verification (CAV) (July 2020)
47. Usman, M., Gopinath, D., Sun, Y., Noller, Y., Păsăreanu, C.S.: Nnrepair: Constraint-based repair of neural network classifiers. In: Silva, A., Leino, K.R.M. (eds.) Computer Aided Verification. pp. 3–25. Springer International Publishing, Cham (2021)
48. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). pp. 1599–1614 (2018)
49. 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. Advances in Neural Information Processing Systems **34** (2021)
50. Xiao, K.Y., Tjeng, V., Shafiullah, N.M.M., Madry, A.: Training for faster adversarial robustness verification via inducing reLU stability. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=BJfIVjAcKm
51. Yang, X., Yamaguchi, T., Tran, H.D., Hoxha, B., Johnson, T.T., Prokhorov, D.: Neural network repair with reachability analysis. In: Bogomolov, S., Parker, D. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 221–236. Springer International Publishing, Cham (2022)

A Appendix



	Name	Type	Activations
1	imageinput 28×28×1 images with 'zerocenter' norm…	Image Input	28(S) × 28(S) × 1(C) × 1(B)
2	conv 3 3×3×1 convolutions with stride [1 1] an…	2-D Convolution	26(S) × 26(S) × 3(C) × 1(B)
3	batchnorm Batch normalization with 3 channels	Batch Normalization	26(S) × 26(S) × 3(C) × 1(B)
4	relu ReLU	ReLU	26(S) × 26(S) × 3(C) × 1(B)
5	avgpool2d 2×2 average pooling with stride [2 2] an…	2-D Average Pooling	13(S) × 13(S) × 3(C) × 1(B)
6	flatten Flatten	Flatten	507(C) × 1(B)
7	fc 10 fully connected layer	Fully Connected	10(C) × 1(B)
8	softmax softmax	Softmax	10(C) × 1(B)
9	classoutput crossentropyex with '0' and 9 other clas…	Classification Output	10(C) × 1(B)

Fig. 9. Neural network architecture of the models trained on MNIST. The architecture for the MedNIST models is created with the same parameters as the MNIST ones, but the difference is the size of the input. MNIST input is 28×28, while MedNIST images are 64×64, thus leading to small differences such as in the number of weights across some of the layers.

Table 1. MedNIST summary results

Model			**Accuracy (%)**	**Verification Results**			
Regularization	*Initialization*	*Seed*		*Robust*	*Unkown*	*Not Robust*	*Avg. Time (s)*
Dropout	Glorot	0	99.27	276	24	0	28.87
Dropout	Glorot	1	99.51	294	6	0	13.33
Dropout	Glorot	2	99.52	285	15	0	11.93
Dropout	Glorot	3	99.57	257	43	0	24.53
Dropout	Glorot	4	99.63	286	14	0	17.36
Dropout	He	0	99.65	257	43	0	43.81
Dropout	He	1	99.60	286	14	0	22.23
Dropout	He	2	99.57	203	96	0	28.40
Dropout	He	3	99.73	235	65	0	42.11
Dropout	He	4	99.70	196	104	0	43.95
Dropout	Narrow-normal	0	99.55	300	0	0	10.49
Dropout	Narrow-normal	1	99.50	294	6	0	11.72
Dropout	Narrow-normal	2	99.61	293	7	0	14.85
Dropout	Narrow-normal	3	99.55	296	4	0	5.17
Dropout	Narrow-normal	4	99.66	300	0	0	9.35
Jacobian	Glorot	0	99.79	288	12	0	19.55
Jacobian	Glorot	1	99.79	291	9	0	20.80
Jacobian	Glorot	2	99.80	289	11	0	19.97
Jacobian	Glorot	3	99.80	289	11	0	19.98
Jacobian	Glorot	4	99.78	292	8	0	21.76
Jacobian	He	0	99.67	300	0	0	14.46
Jacobian	He	1	99.75	298	2	0	14.72
Jacobian	He	2	99.74	298	2	0	14.85
Jacobian	He	3	99.74	298	2	0	14.81
Jacobian	He	4	99.77	297	3	0	15.00
Jacobian	Narrow-normal	0	99.69	300	0	0	10.32
Jacobian	Narrow-normal	1	99.76	295	5	0	9.51
Jacobian	Narrow-normal	2	99.77	300	0	0	10.33
Jacobian	Narrow-normal	3	99.78	300	0	0	10.32
Jacobian	Narrow-normal	4	99.78	300	0	0	10.31
L_2	Glorot	0	99.31	239	61	0	38.67
L_2	Glorot	1	99.51	292	8	0	18.25
L_2	Glorot	2	99.55	273	27	0	16.24
L_2	Glorot	3	99.60	229	71	0	32.63
L_2	Glorot	4	99.65	277	22	1	23.67
L_2	He	0	99.69	260	39	1	45.94
L_2	He	1	99.57	292	8	0	27.76
L_2	He	2	99.58	196	104	0	39.09
L_2	He	3	99.72	274	26	0	42.45
L_2	He	4	99.72	188	112	0	44.44
L_2	Narrow-normal	0	99.71	294	6	0	35.24
L_2	Narrow-normal	1	99.53	292	8	0	10.52
L_2	Narrow-normal	2	99.61	293	7	0	17.80
L_2	Narrow-normal	3	99.55	298	2	0	9.21
L_2	Narrow-normal	4	99.68	299	1	0	12.43

Table 2. MNIST summary results

Model			**Accuracy (%)**	**Verification Results**			
Regularization	*Initialization*	*Seed*		*Robust*	*Unkown*	*Not Robust*	*Avg. Time (s)*
Dropout	Glorot	0	95.61	300	0	0	4.15
Dropout	Glorot	1	95.54	299	1	0	3.87
Dropout	Glorot	2	93.43	293	7	0	0.32
Dropout	Glorot	3	96.56	298	2	0	3.90
Dropout	Glorot	4	94.10	298	2	0	0.41
Dropout	He	0	92.79	297	3	0	4.04
Dropout	He	1	93.62	299	1	0	0.32
Dropout	He	2	93.82	299	1	0	0.26
Dropout	He	3	95.97	299	1	0	4.19
Dropout	He	4	95.81	299	0	1	4.05
Dropout	Narrow-normal	0	95.92	300	0	0	0.52
Dropout	Narrow-normal	1	93.31	299	1	0	0.32
Dropout	Narrow-normal	2	94.67	299	1	0	0.38
Dropout	Narrow-normal	3	96.03	300	0	0	4.53
Dropout	Narrow-normal	4	95.84	300	0	0	0.48
Jacobian	Glorot	0	96.37	299	1	0	4.65
Jacobian	Glorot	1	96.86	298	2	0	6.11
Jacobian	Glorot	2	97.03	299	1	0	6.32
Jacobian	Glorot	3	97.16	296	4	0	5.58
Jacobian	Glorot	4	97.08	299	1	0	5.36
Jacobian	He	0	94.81	298	2	0	5.89
Jacobian	He	1	96.04	297	3	0	3.93
Jacobian	He	2	96.56	299	1	0	2.57
Jacobian	He	3	96.97	299	1	0	1.75
Jacobian	He	4	96.83	296	4	0	1.57
Jacobian	Narrow-normal	0	96.50	300	0	0	6.08
Jacobian	Narrow-normal	1	97.12	299	1	0	3.46
Jacobian	Narrow-normal	2	96.97	300	0	0	2.53
Jacobian	Narrow-normal	3	97.23	299	1	0	1.96
Jacobian	Narrow-normal	4	97.20	299	1	0	1.98
L_2	Glorot	0	96.11	300	0	0	3.64
L_2	Glorot	1	95.63	296	4	0	7.00
L_2	Glorot	2	94.49	293	7	0	0.30
L_2	Glorot	3	96.29	295	5	0	7.08
L_2	Glorot	4	94.49	299	1	0	0.42
L_2	He	0	92.88	300	0	0	3.54
L_2	He	1	93.73	295	5	0	3.60
L_2	He	2	93.47	300	0	0	0.27
L_2	He	3	96.03	296	4	0	3.72
L_2	He	4	96.18	300	0	0	3.67
L_2	Narrow-normal	0	96.20	299	1	0	3.72
L_2	Narrow-normal	1	93.27	297	3	0	0.29
L_2	Narrow-normal	2	94.97	299	1	0	3.72
L_2	Narrow-normal	3	96.68	299	1	0	7.02
L_2	Narrow-normal	4	95.72	298	2	0	3.70