Interpreting Robustness Proofs of Deep Neural Networks

Debangshu Banerjee [1] **Avaljot Singh** [1] **Gagandeep Singh** [1 2]

Abstract

Numerous methods have emerged to verify the robustness of deep neural networks (DNNs). While effective in providing theoretical guarantees, the proofs generated using these techniques often lack human interpretability. Our paper bridges this gap by introducing new concepts, algorithms, and representations that generate human-understandable interpretations of the proofs. Using our approach, we discover that standard DNN proofs rely more on irrelevant input features compared to provably robust DNNs. Provably robust DNNs filter out spurious input features, but sometimes it comes at the cost of semantically meaningful ones. DNNs combining adversarial and provably robust training strike a balance between the two. Overall, our work enhances human comprehension of proofs and sheds light on their reliance on different types of input features.

1. Introduction

Lack of trust stemming from the black-box behaviors of DNNs poses a major challenge in their real-world deployment. To mitigate this issue there has been substantial work on interpreting individual DNN predictions to understand their internal workings. However, existing DNN interpretation methods (Sundararajan et al., 2017) fail to provide human-understandable insights for infinite input sets. Orthogonally, the field of DNN verification has emerged to provide formal guarantees on DNN behaviors on an infinite set of inputs. DNN verifiers (Singh et al., 2019c; Zhang et al., 2018) generate formal proofs sufficient to prove network robustness but it is not clear whether they are based on any meaningful input features learned by the DNN that are necessary for correct classification. So to improve trust, we propose for the first time, the problem of interpreting the invariants captured by DNN robustness proofs.

Key Challenges and contributions. State-of-the-art DNN verifiers generate proofs involving high-dimensional convex shapes across thousands of neurons. Interpreting these shapes is challenging, so we aim to identify the most important parts of the proof. This requires defining importance and developing methods to identify them. We make the following contributions to overcome these challenges:

- We introduce a novel concept of proof features that can be analyzed independently by generating the corresponding interpretations. We propose a priority function over them that signifies their importance in the complete proof.
- We design a general algorithm - ***ProFIt*** (**Pro**of **F**eature **Int**erpretation) that extracts a set of important proof features that preserve the proof.
- We compare interpretations of the proof features for standard DNNs and state-of-the-art robustly trained DNNs for the MNIST and CIFAR10 datasets. We observe that the proof features corresponding to the standard networks rely on spurious input features while the proofs of adversarially trained DNNs (Madry et al., 2018) filter out some of the spurious features. In contrast, the networks trained with certifiable training (Zhang et al., 2020) produce proofs that do not rely on any spurious features but they also miss out on some meaningful features. Proofs for training methods that combine both empirical and certified robustness (Balunovic and Vechev, 2020) provide a common ground. We empirically show that these observations are not contingent on any specific DNN verifier.

2. Preliminaries

DNN verification. DNN verification involves proving that the network outputs $Y = N(X)$ corresponding to all inputs X from an input region specified by ϕ, satisfy a logical linear specification ψ. The output specification, is written as $\psi(Y) = (C^T Y \geq 0)$ where $C \in \mathbb{R}^{d_l}$ defines the linear inequality for encoding the robustness property. Throughout the paper, we refer to the input region ϕ and output specification ψ together as *property* (ϕ, ψ).
A DNN verifier $\mathcal{V}$ symbolically computes a possibly over-approximated output region $\mathcal{A} \subseteq \mathbb{R}^{d_l}$ containing all possible outputs of N corresponding to ϕ. Let $\Lambda(\mathcal{A}) = \min_{Y \in \mathcal{A}} C^T Y$ denote the minimum value of $C^T Y$ where $Y \in \mathcal{A}$. Then N satisfies property (ϕ, ψ) if $\Lambda(\mathcal{A}) \geq 0$. In

[*]Equal contribution [1]Department of Computer Science, University of Illinois, Urbana-Champaign, USA [2]VMware Research, Palo Alto, USA. Correspondence to: Debangshu Banejee <db21@illinois.edu>.

2^{nd} *Workshop on Formal Verification of Machine Learning*, Honolulu, Hawaii, USA. Colocated with ICML 2023. Copyright 2023 by the author(s).

this paper, we primarily focus on deterministic, sound, and incomplete verifiers (Singh et al., 2018; Zhang et al., 2018) over complete verifiers for their lower computational cost.

DNN interpretation with sparse decision layer. DNNs considered in this paper, have complex multi-layer structures, making them harder to interpret. Instead of interpreting what each layer of the network is doing, recent works (Wong et al., 2021; Liao and Cheung, 2022) treat DNNs as the composition of a *deep feature extractor* and an affine *decision layer*. The output at each neuron of the penultimate layer represents a single deep feature and the final affine layer linearly combines them to compute the network output. This perspective enables us to identify the set of features used by the network to compute its output and to investigate their semantic meaning using the existing feature visualization techniques (Ribeiro et al., 2016; Simonyan et al., 2014). However, visualizing each feature is impractical for large DNNs where the penultimate layer can contain thousands of neurons. To address this, the work (Wong et al., 2021) tries to identify a smaller set of features sufficient to maintain the performance of the network. This smaller but sufficient feature set retains only the most important features corresponding to a given input. It is shown empirically (Wong et al., 2021) that a subset of these features of size less than 10 is sufficient to maintain their accuracy.

3. Interpreting DNN Proofs

Proof features. Similar to existing works (Wong et al., 2021), for proof interpretation, we propose to segregate the final decision layer from the network and focus at the features extracted at the penultimate layer. For a given input region ϕ, we look at the symbolic shape (for example - intervals, zonotopes, polytopes, etc.) computed by the verifier at the penultimate layer and then compute its projection on each dimension of the penultimate layer. The projection is an interval $[l_n, u_n]$ which contains all possible output values of the corresponding neuron n with respect to ϕ.

Definition 1 (Proof Features)**.** *Given a network N, input region ϕ and neural network verifier $\mathcal{V}$, for each neuron n_i at the penultimate layer of N, the proof feature $\mathcal{F}_{n_i}$ extracted at that neuron n_i is an interval $[l_{n_i}, u_{n_i}]$ such that $\forall X \in \phi$, the output at n_i always lies in the range $[l_{n_i}, u_{n_i}]$.*

We use $\mathcal{F}$ to denote the set of all proof features at the penultimate layer and $\mathcal{F}_S$ to denote the proof features corresponding to $S \subseteq [d_{l-1}]$ i.e. $\mathcal{F}_S = \{\mathcal{F}_{n_i} \mid i \in S\}$. If N is formally verified by the verifier $\mathcal{V}$ to satisfy the property (ϕ, ψ), then in order to gain insights about the proof generated by $\mathcal{V}$, we can directly investigate (described in section A.2) all the proof features $\mathcal{F}$. However, contemporary networks can have thousands of proof features, but many of them may be spurious and unimportant. So, we aim to identify a smaller, more important set of proof features that satisfy the property (ϕ, ψ). The key challenge lies in defining the most important set of proof features w.r.t the property (ϕ, ψ).

3.1. Sufficient Proof Features

We argue that an important proof feature set, $\mathcal{F}_{S_0} \subseteq \mathcal{F}$, must prove the property (ϕ, ψ) with verifier $\mathcal{V}$ and must have the minimum number of proof features necessary to satisfy the property. The minimality of $\mathcal{F}_{S_0}$ ensures only the proof features essential for proving the property are retained. This hypothesis allows us to view the extraction of important proof features as finding a minimum proof feature set that preserves the property (ϕ, ψ) with verifier $\mathcal{V}$.

Definition 2 (Sufficient proof feature set)**.** *For the proof of property (ϕ, ψ) on DNN N with verifier $\mathcal{V}$, a nonempty set $\mathcal{F}_S \subseteq \mathcal{F}$ of proof features is sufficient if the verifier $\mathcal{V}$ proves the property by using only the proof features in $\mathcal{F}_S$.*

Definition 3 (Minimum proof feature set)**.** *A minimum proof feature set $\mathcal{F}_{S_0} \subseteq \mathcal{F}$ for DNN N verified with $\mathcal{V}$ on (ϕ, ψ) is a sufficient proof feature set having the minimum size.*

Extracting a minimum set of proof features $\mathcal{F}_{S_0}$ from $\mathcal{F}$ is equivalent to pruning maximum number of proof features from $\mathcal{F}$ without violating the property (ϕ, ψ). Let, $W_l[:, i] \in \mathbb{R}^{d_l}$ denote the i-th column of the weight matrix W_l of the final network layer N_l. Pruning any proof feature $\mathcal{F}_{n_i}$ results in setting all weights in $W_l[:, i]$ to 0. Therefore, to compute $\mathcal{F}_{S_0}$, it is sufficient to devise an algorithm that can prune maximum number of columns from W_l while still preserving the property (ϕ, ψ). For any proof feature set $\mathcal{F}_S \subseteq \mathcal{F}$, let $W_l(S) \in \mathbb{R}^{d_l \times d_{l-1}}$ be the weight matrix of the pruned final layer that only retains proof features corresponding to $\mathcal{F}_S$. Then column $W_l(S)[:, i] = W_l[:, i]$ if $i \in S$ and $\underline{0}$ otherise where $\underline{0} \in \mathbb{R}^{d_{l-1}}$ dentoes a constant all-zero vector. The proof feature set $\mathcal{F}_S$ is sufficient iff the property (ϕ, ψ) can be verified by $\mathcal{V}$ on N with the pruned weight matrix $W_l(S)$. As described in Section 2, for property verification the verifier $\mathcal{V}$ computes an over-approximated output region $\mathcal{A}$ of N over the input region ϕ. Let $\mathcal{A}(W_l, S)$ denote the over-approximated output region corresponding to $W_l(S)$. The neural network N can be verified by $\mathcal{V}$ on the property (ϕ, ψ) with $W_l(S)$ iff the lower bound $\Lambda(\mathcal{A}(W_l, S)) \geq 0$. Therefore, finding S_0 corresponding to a minimum proof feature set $\mathcal{F}_{S_0}$ can be formulated as below where for any $S \subseteq [d_{l-1}]$, $|S|$ denotes the number of elements in S.

$$\underset{S \neq \emptyset,\ S \subseteq [d_{l-1}]}{\arg\min} \quad |S| \quad \text{s.t.} \quad \Lambda(\mathcal{A}(W_l, S)) \geq 0 \tag{1}$$

3.2. Approximate Minimum Proof Feature Extraction

The search space for finding $\mathcal{F}_{S_0}$ is prohibitively large and contains $2^{d_{l-1}}$ possible candidates. So, computing a minimum solution with an exhaustive search is infeasible. We design a practically efficient approximation algorithm based on

Table 1. ProFIt Efficacy Analysis

Dataset	Training Method	Input Region (ϕ) eps (ϵ)	No. of proved properties	Original Feature Count	Mean Proof Feature Count			No. of proofs with ≤ 5 proof features (ProFIt)	No. of proofs with ≤ 10 proof features (ProFIt)
					Random	Gradient	ProFIt		
MNIST	Standard	0.02	459	100	20.31	5.25	1.96	449	457
	PGD Trained	0.02	415	1000	93.29	13.73	6.02	315	364
	COLT	0.02	480	100	14.45	5.43	3.46	401	461
	CROWN-IBP	0.02	482	100	9.51	6.73	6.16	240	401
CIFAR-10	Standard	0.2/255	277	100	30.36	18.28	11.12	127	173
	PGD Trained	0.2/255	298	100	31.22	16.58	9.74	173	210
	COLT	0.2/255	267	250	30.10	18.13	9.03	170	204
	CROWN-IBP	0.2/255	265	256	7.96	7.49	5.30	172	221

a greedy heuristic that can generate a small (may not always be minimum) sufficient feature set with only $O(\log(d_{l-1}))$ verifier calls. The detailed algorithm is described in appendix A. At a high level, the heuristic attempts to estimate whether pruning each proof feature $\mathcal{F}_{n_i}$ from a sufficient feature set violates the property (ϕ, ψ) or not. Subsequently, we prioritize pruning those proof features $\mathcal{F}_{n_i}$ that, if pruned, will likely preserve the proof of the property (ϕ,ψ) with the verifier $\mathcal{V}$. Given a sufficient proof features set $\mathcal{F}_S$, for a proof feature $\mathcal{F}_{n_i} \in \mathcal{F}_S$, we compute its importance by estimating the change $\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S)$ that occurs to $\Lambda(\mathcal{A}(W_l, S))$ if $\mathcal{F}_{n_i}$ is pruned from $\mathcal{F}_S$. Let, the over-approximated output region computed by verifier $\mathcal{V}$ corresponding to $\mathcal{F}_S \setminus \{\mathcal{F}_{n_i}\}$ be $\mathcal{A}(W_l, S \setminus \{i\})$ then $\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S)$ is defined as follows

$$\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S) = |\Lambda(\mathcal{A}(W_l, S)) - \Lambda(\mathcal{A}(W_l, S \setminus \{i\}))|$$

However, $\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S)$ depends on the specific $\mathcal{F}_S$ and doesn't estimate the global importance of $\mathcal{F}_{n_i}$. To address this, we define the priority $P(\mathcal{F}_{n_i})$ as the maximum $\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S)$ over all sufficient feature set $\mathcal{F}_S$ containing $\mathcal{F}_{n_i}$. Since iterating over all $\mathcal{F}_S$ is impractical, we compute an upper bound $P_{ub}(\mathcal{F}_{n_i})$ for $P(\mathcal{F}_{n_i})$ that holds for all $\mathcal{F}_S$. This global upper bound enables efficient computation without iterating over all $\mathcal{F}_S$. For the network N and input region ϕ, let $\mathcal{A}_{l-1}$ denote the over-approximate region computed by $\mathcal{V}$ at the penultimate layer. Then, the global uppper bound of $\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S)$ can be computed as follows where for any vector $X \in \mathbb{R}^{d_{l-1}}$, x_i denotes its i-th coordinate:

$$\Delta(\mathcal{F}_{n_i}, \mathcal{F}_S) \leq \max_{X \in \mathcal{A}_{l-1}} |(C^T W_l(S)X - C^T W_l(S \setminus \{i\})X)|$$

$$P(\mathcal{F}_{n_i}) = \max_{\mathcal{F}_S} \Delta(\mathcal{F}_{n_i}, \mathcal{F}_S) \leq \max_{X \in \mathcal{A}_{l-1}} |(C^T W_l[:, i]) \cdot x_i)|$$

Now, any proof feature $\mathcal{F}_{n_i} = [l_{n_i}, u_{n_i}]$ computed by $\mathcal{V}$ contains all possible values of x_i where $X \in \mathcal{A}_{l-1}$. Leveraging this observation, we can further simplify the upper bound $P_{ub}(\mathcal{F}_{n_i})$ of $P(\mathcal{F}_{n_i})$ as shown below. This simplification ensures that $P_{ub}(\mathcal{F}_{n_i})$ for all $\mathcal{F}_{n_i}$ can be computed with $O(d_{l-1})$ elementary vector operations.

$$P_{ub}(\mathcal{F}_{n_i}) = |(C^T W_l[:, i])| \cdot \max(|l_{n_i}|, |u_{n_i}|) \qquad (2)$$

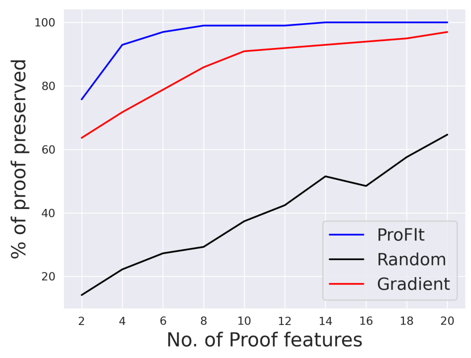
4. Experimental Evaluation

4.1. Efficacy of ProFIt Algorithm

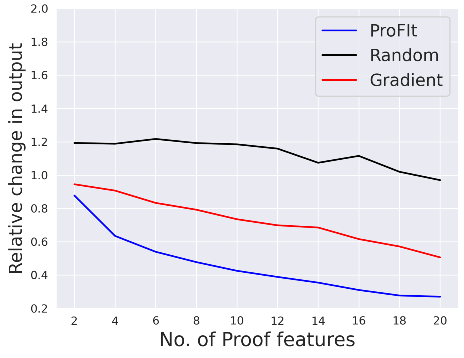
In this section, we evaluate the effectiveness of ProFIt in extracting sufficient proof feature sets and the usefulness of the proposed priority ordering. We assess the size of the extracted feature sets used for interpretation, as it reflects the ease of interpreting the proof. We compare ProFIt with two other priority heuristics: (i) a random ordering of proof features and (ii) sorting proof features based on gradient magnitudes computed w.r.t the verifier output. For each network, we use the first 500 images from their corresponding test sets for defining input specification ϕ. We conduct experiments with two different ϵ values for defining L_∞ input region. The ϵ values used for MNIST networks are 0.02 and 0.1 and that for CIFAR-10 networks are $0.2/255$ and $2/255$. For experiments with high ϵ values (0.1 for MNIST and $2/255$ for CIFAR-10), we omit standard DNNS as they do not satisfy robustness properties defined with larger ϵ. Unless specified otherwise, we use state-of-the-art incomplete verifier α-Crown (Xu et al., 2021) for all our experiments.

Mean proof feature set size comparison: In Table 1, we summarize the results of the experiments for different networks. We show that the mean size of the proof feature set computed using ProFIt (column 8) is significantly smaller than that of the random and gradient-based baselines (columns 6 and 7 respectively). Detailed results are in Appendix D.1 and D.2.

Evaluation of proof feature priority: Next, we evaluate the efficacy of the priority ordering of proof features defined in Eq. 2 against the random and gradient-based priority ordering defined above. For these experiments, we fix the proof feature set size and compare the extracted proof feature sets with the fixed size based on two metrics - (i) %



(a) Percentages of proof preserved



(b) Mean relative change in verifier output

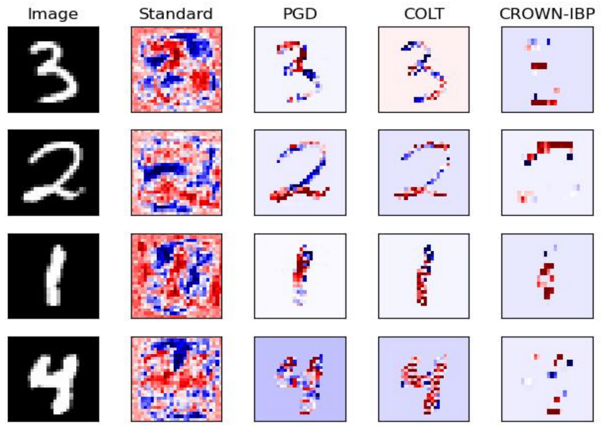
Figure 1. Efficacy analysis of the priority heuristic on PGD-trained MNIST networks.

cases the extracted proof feature set preserves the property (ϕ, ψ) that was satisfied initially and (ii) mean relative change in verifier output after every proof feature, not part of the extracted set is removed. Since we fix the extracted proof feature set size, the feature set is no longer guaranteed to be sufficient. For proof feature set size varying from 2 to 20, we show that the priority ordering used by ProFIt preserves a higher % of proofs (Fig. 1a) while also achieving a lower relative change from the original verifier output (Fig. 1b) compared to both the baselines. These plots are generated on PGD-trained MNIST networks for $\epsilon = 0.1$. Plots for other MNIST networks and CIFAR-10 networks are shown in Appendix D.3. Additionally, in Appendix D.4 we provide a qualitative evaluation of the priority ordering of proof features where we show that interpretations of proof features with higher priority capture meaningful input features while interpretations of proof features with lower priority are less informative.

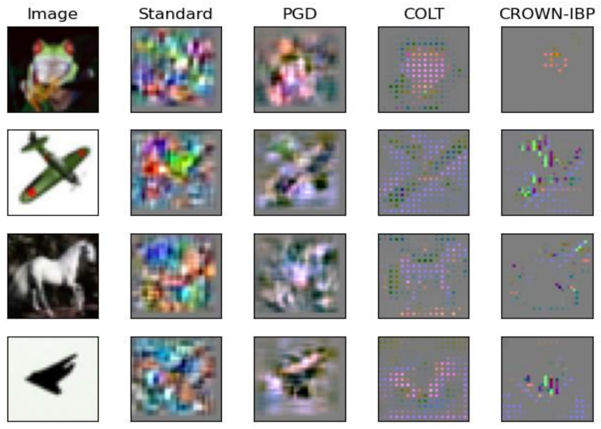
4.2. Qualititive comparison of robustness proofs

In this section, we interpret proof features obtained with ProFIt and use these interpretations to qualitatively evaluate different proofs of the same robustness property generated on standard and robustly trained networks with the same verifier - α-Crown. We also study the effect of certified robust training methods like CROWN-IBP (Zhang et al., 2020), empirically robust training methods like PGD (Madry et al., 2018) and training methods that combine both adversarial and certified training like COLT (Balunovic and Vechev, 2020) on the proof features. For an input region ϕ, we say that a robustness proof is semantically meaningful if it focuses on the relevant features of the output class for images contained inside ϕ. In the case of MNIST or CIFAR-10 images, spurious features are the pixels that form a part of the background of the image, whereas meaningful features are the pixels that are a part of the actual object being identified by the network. Gradient map of the extracted proof features w.r.t. to the input region ϕ gives us an idea of the input pixels that the network focuses on. We obtain the gradient maps by calculating the mean gradient over 500 uniformly drawn samples from ϕ as described in Appendix A.2.

In Fig. 2, we compare the gradient maps corresponding to the top proof feature (the one having the highest priority $P_{ub}(\mathcal{F}_{n_i})$) on networks from Table 1 on representative images of different output classes in the MNIST and CIFAR10 test sets. These experiments lead us to interesting observations - even if some property is verified for both the standard network and the robustly trained network, there is a difference in the human interpretability of the types of input features that the proofs rely on. The standard networks and the provably robust trained networks like CROWN-IBP are the two extremes of the spectrum. For the networks obtained with standard training, we observe that although the top-proof feature does depend on some of the semantically meaningful regions of the input image, the gradient at several spurious features is also non-zero. On the other hand, the top proof feature corresponding to state-of-the-art provably robust training method CROWN-IBP filters out most of the spurious features, but it also misses out on some meaningful features. The proofs of PGD-trained networks filter out the spurious features and are, therefore, more semantically aligned than the standard networks. The proofs of the training methods that combine both empirical robustness and provable robustness like COLT in a way provide the best of both worlds by not only selectively filtering out the spurious features but also highlighting the more human interpretable features, unlike the certifiably trained networks. So, as the training methods tend to regularize more for robustness, their proofs become more selective in the type of input features that they rely on. To further support our observation, we show additional plots for the top proof feature visualization in Appendix D.5 and visualization for multiple proof features in Appendix D.6. The extracted proof features set and their gradient maps computed w.r.t high ϵ



(a) Gradient maps generated on MNIST networks.



(b) Gradient maps generated on CIFAR-10 networks.

Figure 2. Gradient map corresponding to the top proof feature corresponding to DNNs trained using different methods rely on different input features.

values ($\epsilon = 0.1$ for MNIST and $\epsilon = 2/255$ for CIFAR-10) are similar to those generated with smaller ϵ as shown in Appendix D.7. In Appendix E, we empirically show that these observations are not contingent on the verifier - α-Crown.

5. Conclusion

In this work, we develop a novel method called ProFIt to interpret neural network robustness proofs. We empirically establish that even if a property holds for a DNN, the proof for the property may rely on spurious or semantically meaningful features depending on the training method used to train the DNNs. We believe that ProFIt can be applied for diagnosing the trustworthiness of DNNs inside their development pipeline.

References

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 2020.

Stanley Bak, Hoang-Dung Tran, Kerianne Hobbs, and Taylor T. Johnson. Improved geometric path enumeration for verifying relu neural networks. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, pages 66–96. Springer, 2020. doi: 10.1007/978-3-030-53288-8_4. URL https://doi.org/10.1007/978-3-030-53288-8_4.

Mislav Balunovic and Martin T. Vechev. Adversarial training and provable defenses: Bridging the gap. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Pushmeet Kohli, P Torr, and P Mudigonda. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(2020), 2020a.

Rudy R Bunel, Oliver Hinder, Srinadh Bhojanapalli, and Krishnamurthy Dvijotham. An efficient nonconvex reformulation of stagewise convex optimization problems. *Advances in Neural Information Processing Systems*, 33, 2020b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/cohen19c.html.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, 2017.

Claudio Ferrari, Mark Niklas Mueller, Nikola Jovanović, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=l_amHf1oaK.

Aymeric Fromherz, Klas Leino, Matt Fredrikson, Bryan Parno, and Corina Pasareanu. Fast geometric projections for local robustness certification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zWy1uxjDdZJ.

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.

Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019.

Cheng-Yu Hsieh, Chih-Kuan Yeh, Xuanqing Liu, Pradeep Kumar Ravikumar, Seungyeon Kim, Sanjiv Kumar, and Cho-Jui Hsieh. Evaluations and methods for explanation through robustness analysis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=4dXmpCDGNp7.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1989.

Zukang Liao and Michael Cheung. Automated invariance testing for machine learning models using sparse linear layers. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*, 2022. URL https://openreview.net/forum?id=VP8ATzLGyQx.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. International Conference on Learning Representations (ICLR)*, 2018.

Denis Mazzucato and Caterina Urban. Reduced products of abstract domains for fairness certification of neural networks. In Cezara Dragoi, Suvam Mukherjee, and Kedar S. Namjoshi, editors, *Static Analysis - 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17-19, 2021, Proceedings*, volume 12913 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2021. doi: 10.1007/978-3-030-88806-0_15. URL https://doi.org/10.1007/978-3-030-88806-0_15.

Alessandro De Palma, Harkirat S. Behl, Rudy R. Bunel, Philip H. S. Torr, and M. Pawan Kumar. Scaling the convex barrier with active sets. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6034.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 31, 2018.

Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. Beyond the single neuron convex barrier for neural network certification. In *Advances in Neural Information Processing Systems*, 2019a.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL), 2019b.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. Robustness certification with refinement. In *International Conference on Learning Representations*, 2019c. URL https://openreview.net/forum?id=HJgeEh09KQ.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL http://arxiv.org/abs/1706.03825.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021a.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021b. URL https://openreview.net/forum?id=ahYIlRBeCFw.

Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning*, pages 11205–11216. PMLR, 2021.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. 2020.

Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nVZtXBI6LNn.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/zhang19p.html.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, 2018.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *Proc. International Conference on Learning Representations, ICLR*, 2020.

Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=5haAJAcofjc.

A. ProFIt algorithm

A.1. Proof feature extraction

Next, we describe how we compute an approximate proof feature set using the feature priorities $P_{ub}(\mathcal{F}_{n_i})$. For any proof feature $\mathcal{F}_{n_i}$, $P_{ub}(\mathcal{F}_{n_i})$ estimates the importance of $\mathcal{F}_{n_i}$ in preserving the proof. So, a trivial step is to just prune all the proof features from $\mathcal{F}$ whose P_{ub} is 0. These features do not have any contribution to the proof of the property (ϕ, ψ) by the verifier $\mathcal{V}$. This step forms a trivial algorithm. However, this is not enough. We can further prune some more proof features leading to a yet smaller set. For this, we propose an iterative algorithm **ProFIt** shown in Algorithm 1 which maintains two set namely, $\boldsymbol{F}_{S_0}$ and $\boldsymbol{F}_S$. $\boldsymbol{F}_{S_0}$ contains the features guaranteed to be included in the final answer computed by ProFIt and $\boldsymbol{F}_S$ contains the candidate features yet to be pruned by the algorithm. At every step, the algorithm ensures that the set $\boldsymbol{F}_S \cup \boldsymbol{F}_{S_0}$ is sufficient and iteratively reduces its size by pruning proof features from $\boldsymbol{F}_S$. The algorithm iteratively prunes the feature $\mathcal{F}_{n_i}$ with the lowest value of $P_{ub}(\mathcal{F}_{n_i})$ from $\boldsymbol{F}_S$ to maximize the likelihood that $\boldsymbol{F}_S \cup \boldsymbol{F}_{S_0}$ remains sufficient at each step. At Line 8 in the algorithm, $\boldsymbol{F}_{S_0}$ and $\boldsymbol{F}_S$ are initialized as empty set ($\{\}$) and $\mathcal{F}$ respectively. Removing a single feature in each iteration and checking the sufficiency of the remaining features in the worst case leads to $O(d_{l-1})$ incremental verification calls which are expensive. Instead, at each step, from $\boldsymbol{F}_S$ our algorithm greedily picks top-$|S|/2$ features (line 10) $\boldsymbol{F}_{S_1}$ based on their priority and invokes the verifier $\mathcal{V}$ to check the sufficiency of $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ (line 12). If the feature set $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ is sufficient (line 13), ProFIt removes all features in $\boldsymbol{F}_S \setminus \boldsymbol{F}_{S_1}$ from $\boldsymbol{F}_S$ and therefore $\boldsymbol{F}_S$ is updated as $\boldsymbol{F}_{S_1}$ in this step (line 14). Otherwise, if $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ does not preserve the property (ϕ,ψ) (line 15), ProFIt adds all feature in $\boldsymbol{F}_{S_1}$ to $\boldsymbol{F}_{S_0}$ (line 16) and replaces $\boldsymbol{F}_S$ with $\boldsymbol{F}_S \setminus \boldsymbol{F}_{S_1}$ (line 17). The algorithm terminates after all features in $\boldsymbol{F}_S$ are exhausted. Since at every step, the algorithm reduces size of $\boldsymbol{F}_S$ by half, it always terminates within $O(\log(d_{l-1}))$ incremental verifier calls. In Appendix B, we derive mathematical guarantees about the correctness and efficacy of Algorithm 1. For correctness, we prove that the feature set $\boldsymbol{F}_{S_0}$ is always sufficient (Definition 2). For efficacy, we theoretically find a non-trivial upper bound on the size of $\boldsymbol{F}_{S_0}$.

Algorithm 1 Approx. minimum proof feature computation

```
 1: Input: DNN $N$, property $(\phi, \psi)$, verifier $\mathcal{V}$.
 2: Output: approx. minimum proof features $\boldsymbol{F}_{S_0}$,
 3: if $\mathcal{V}$ can not verify $N$ on $(\phi, \psi)$ then
 4:     return
 5: end if
 6: Calculate all proof features for input region $\phi$.
 7: Calculate priority $P_{ub}(\mathcal{F}_{n_i})$ all proof features.
 8: Initialization: $\boldsymbol{F}_{S_0} = \{\}$, $\boldsymbol{F}_S = \mathcal{F}$
 9: while $\boldsymbol{F}_S$ is not empty do
10:     $\boldsymbol{F}_{S_1}$ = top-$|S|/2$ features based on $P_{ub}(\mathcal{F}_{n_i})$
11:     $\boldsymbol{F}_{S_2} = \boldsymbol{F}_S \setminus \boldsymbol{F}_{S_1}$
12:     Check sufficiency of $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ with $\mathcal{V}$
13:     if $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ is sufficient then
14:         $\boldsymbol{F}_S = \boldsymbol{F}_{S_1}$
15:     else
16:         $\boldsymbol{F}_{S_0} = \boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$
17:         $\boldsymbol{F}_S = \boldsymbol{F}_{S_2}$
18:     end if
19: end while
20: return proof features $\boldsymbol{F}_{S_0}$.
```

A.2. Visualization of extracted proof features

To interpret DNN robustness proofs, we analyze the semantic meaning of extracted features. We adapt existing local visualization techniques (Sundararajan et al., 2017; Smilkov et al., 2017) to estimate the relevance of proof features within an input region ϕ. Using a large, uniformly drawn sample from ϕ, we compute $\mathcal{G}(\mathcal{F}_{n_i}, \phi)$ as the mean gradient of n_i's output with respect to inputs, i.e., $\mathcal{G}(\mathcal{F}_{n_i}, \phi) = \mathbb{E}_{X \sim \phi}\, \mathcal{G}(n_i, X)$.

B. Theoretical Guarantees of ProFIt

B.1. Proof of sufficiency of the extracted proof feature set

Theorem 1. *If the verifier $\mathcal{V}$ can prove the property (ϕ, ψ) on the network N, then $\boldsymbol{F}_{S_0}$ computed by Algorithm 1 is sufficient (Definition 2).*

Proof. Proof of theorem 1 by induction on the number of steps of the while loop.
Induction Hypothesis: At each step of the loop, $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_S$ is sufficient.
Base Case: At step 0, i.e., at initialization, $\boldsymbol{F}_{S_0} = \{\}$ and $\boldsymbol{F}_S = \mathcal{F}$. So, $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_S = \mathcal{F}$. Given that $\mathcal{V}$ proves the property (ϕ, ψ) on N, from Definition 2, $\mathcal{F}$ is sufficient.
Induction Case: Let $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_S$ be sufficient for n-th step of the loop. Consider the following cases for $(n+1)$-th step of the loop.

1. Let $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ be sufficient at line 12. In this case, $\boldsymbol{F}_S$ is updated by $\boldsymbol{F}_{S_1}$ (line 14). So, $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_S$ is sufficient.
2. Let $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ be not sufficient at line 12. In this case, $\boldsymbol{F}_{S_0}$ and $\boldsymbol{F}_S$ are updated as in lines 16 and 17. Let the new $\boldsymbol{F}_{S_0}$ and $\boldsymbol{F}_S$ be $\boldsymbol{F}'_{S_0}$ and $\boldsymbol{F}'_S$. So, $\boldsymbol{F}'_{S_0} = \boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ and $\boldsymbol{F}'_S = \boldsymbol{F}_{S_2}$. So, $\boldsymbol{F}'_{S_0} \cup \boldsymbol{F}'_S = \boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1} \cup \boldsymbol{F}_{S_2}$. Also, $\boldsymbol{F}_{S_1} \cup \boldsymbol{F}_{S_2} = \boldsymbol{F}_S$. So, $\boldsymbol{F}'_{S_0} \cup \boldsymbol{F}'_S = \boldsymbol{F}_{S_0} \cup \boldsymbol{F}_S$. So, from induction hypothesis, $\boldsymbol{F}'_{S_0} \cup \boldsymbol{F}'_S$ is sufficient.

□

B.2. Upper bound on the size of the extracted proof feature set

Lemma 1. $\forall S \subseteq [d_{l-1}]$ *if* $i \in S$ *then* $|\Lambda(\mathcal{A}(W_l, S)) - \Lambda(\mathcal{A}(W_l, S \setminus \{i\}))| \leq \max_{X \in \mathcal{A}_{l-1}} |C^T W[:i]X|$.

Proof. Without loss of generality let assume, $\Lambda(\mathcal{A}(W_l, S)) \leq \Lambda(\mathcal{A}(W_l, S \setminus \{i\}))$. Suppose, $\Lambda(\mathcal{A}(W_l, S) = C^T W_l(S) X_{min}$ where $X_{min} \in \mathcal{A}_{l-1}$.

$$\begin{aligned}
\Lambda(\mathcal{A}(W_l, S)) &\leq \Lambda(\mathcal{A}(W_l, S \setminus \{i\})) \\
C^T W_l(S) X_{min} &\leq \Lambda(\mathcal{A}(W_l, S \setminus \{i\})) \leq C^T W_l(S \setminus \{i\}) X_{min} \\
|\Lambda(\mathcal{A}(W_l, S)) - \Lambda(\mathcal{A}(W_l, S \setminus \{i\}))| &= |C^T W_l(S) X_{min} - \Lambda(\mathcal{A}(W_l, S \setminus \{i\}))| \\
&\leq |C^T W_l(S) X_{min} - |C^T W_l(S \setminus \{i\}) X_{min}| \\
&= |C^T W_l[:, i] X_{min}| \\
&\leq \max_{X \in \mathcal{A}_{l-1}} |C^T W_l[:, i] X|
\end{aligned}$$

□

Lemma 2. $\forall \mathcal{F}_S \subseteq \mathcal{F}$, $\delta(\mathcal{F}_S) \leq \sum_{\mathcal{F}_{n_i} \in \mathcal{F} \setminus \mathcal{F}_S} P_{ub}(\mathcal{F}_{n_i})$ *where* $P_{ub}(\mathcal{F}_{n_i})$ *is defined in* (2) *and* $\Delta(\mathcal{F}_S) = |\Lambda(\mathcal{A}) - \Lambda(\mathcal{A}(W_l, S))|$.

Proof.

$$\begin{aligned}
\Delta(\mathcal{F}_S) &= |\Lambda(\mathcal{A}) - \Lambda(\mathcal{A}(W_l, S))| \\
&\leq \max_{X \in \mathcal{A}_{l-1}} |\sum_{\mathcal{F}_{n_i} \in \mathcal{F} \setminus \mathcal{F}_S} C^T W[:i]X| \\
&\leq \max_{X \in \mathcal{A}_{l-1}} \sum_{\mathcal{F}_{n_i} \in \mathcal{F} \setminus \mathcal{F}_S} |C^T W[:i]X| \\
&\leq \sum_{\mathcal{F}_{n_i} \in \mathcal{F} \setminus \mathcal{F}_S} \max_{X \in \mathcal{A}_{l-1}} |C^T W[:i]X| \\
&= \sum_{\mathcal{F}_{n_i} \in \mathcal{F} \setminus \mathcal{F}_S} P_{ub}(\mathcal{F}_{n_i}) \quad [\text{ From } (2)]
\end{aligned}$$

□

Lemma 3. *A feature set $\mathcal{F}_S \subseteq \mathcal{F}$ with $\Delta(\mathcal{F}_S) \leq \Lambda(\mathcal{A})$ is sufficient provided $\Lambda(\mathcal{A}) \geq 0$.*

Proof. $\Delta(\mathcal{F}_S) = |\Lambda(\mathcal{A}) - \Lambda(\mathcal{A}(W_l, S))|$. So, there can be two cases:

1. $\Lambda(\mathcal{A}(W_l, S)) = \Lambda(\mathcal{A}) + \Delta(\mathcal{F}_S)$. Since, $\Lambda(\mathcal{A}) \geq 0$ and $\Delta(\mathcal{F}_S) \geq 0$, $\Lambda(\mathcal{A}(W_l, S)) \geq 0$. So, $\mathcal{F}_S$ is sufficient.
2. $\Lambda(\mathcal{A}(W_l, S)) = \Lambda(\mathcal{A}) - \Delta(\mathcal{F}_S)$
   $\Lambda(\mathcal{A}) \geq 0$ and $\Delta(\mathcal{F}_S) \leq \Lambda(\mathcal{A})$.
   So, $\Lambda(\mathcal{A}(W_l, S)) \geq 0$. So, $\mathcal{F}_S$ is sufficient.

□

Lemma 4. *Let, P_{max} denote the maximum of all priorities $P_{ub}(\mathcal{F}_{n_i})$ over $\mathcal{F}$.*
Let $\mathcal{F}_S \subseteq \mathcal{F}$. If $|\mathcal{F}_S| \leq \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor$, then proof feature set $\mathcal{F}_S^c = \mathcal{F} \setminus \mathcal{F}_S$ is sufficient provided $\Lambda(\mathcal{A}) \geq 0$.

Proof.

$$\begin{aligned} & \forall \mathcal{F}_{n_i} \in \mathcal{F}, P_{ub}(\mathcal{F}_{n_i}) \leq P_{max} \\ \text{From Lemma 2,}\quad & \Delta(\mathcal{F}_S^c) \leq |\mathcal{F}_S| \times P_{max} \\ \text{Also,}\quad & |\mathcal{F}_S| \leq \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor \\ \text{So,}\quad & \Delta(\mathcal{F}_S^c) \leq \Lambda(\mathcal{A}) \\ \text{From Lemma 3,}\quad & \mathcal{F}_S^c \text{ is sufficient.} \end{aligned}$$

□

Definition 4. *Zero proof features set $Z(\mathcal{F})$ denotes the proof features $\mathcal{F}_{n_i} \in \mathcal{F}$ with $P_{ub}(\mathcal{F}_{n_i}) = 0$.*

Theorem 2. *Let, P_{max} denote the maximum of all priorities $P_{ub}(\mathcal{F}_{n_i})$ over $\mathcal{F}$. Given any network N is verified on (ϕ, ψ) with verifier $\mathcal{V}$ then $|\boldsymbol{F}_{S_0}| \leq d_{l-1} - |Z(\mathcal{F})| - \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor$*

Proof of theorem 2. The algorithm 1 arranges the elements of the proof feature set $\mathcal{F}$ in decreasing order according to the priority defined by P_{ub}.
Let $\mathcal{F}'$ be the ordered set corresponding to $\mathcal{F}$. So, $\mathcal{F}' = \mathcal{F}_{n_1} :: \cdots :: \mathcal{F}_{n_m}$, where :: is the list concatenation.
The elements of $Z(\mathcal{F})$ will be at the end of this ordering. So, $\mathcal{F}'$ can be written as $\mathcal{F}'' :: Z(\mathcal{F})$ where $Z(\mathcal{F}) = \mathcal{F}_{n_{k+1}} :: \cdots :: \mathcal{F}_{n_m}$ and $\mathcal{F}'' = \mathcal{F}_{n_1} :: \cdots :: \mathcal{F}_{n_k}$ and p be some of the last elements of $\mathcal{F}''$ s.t. the sum of their priorities just less than $\lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor$, i.e.,

$$\begin{aligned} & p = \mathcal{F}_{n_j} :: \cdots :: \mathcal{F}_{n_k} \\ & \sum_{i=j}^{k} P_{ub}(\mathcal{F}_{n_i}) \leq \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor \\ & \sum_{i=j-1}^{k} P_{ub}(\mathcal{F}_{n_i}) \geq \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor \end{aligned}$$

Further, let $p' = p :: Z(\mathcal{F})$, i.e., $p' = \mathcal{F}_{n_j} :: \cdots :: \mathcal{F}_{n_m}$. Since P_{ub} is 0 for all elements of $Z(\mathcal{F})$,

$$\sum_{i=j}^{m} P_{ub}(\mathcal{F}_{n_i}) \leq \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor \tag{3}$$

Also, $|p'| = |Z(\mathcal{F})| + \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor$ Now, we prove by induction on the number of steps of the while loop in the algorithm 1 that the set $\boldsymbol{F}_{S_0}$ never contains any elements from p'.
Induction Hypothesis: $\boldsymbol{F}_{S_0} \cap p' = \{\}$
Base Case: At initialization, $\boldsymbol{F}_{S_0} = \{\}$. So, the induction hypothesis holds trivially.
Induction Step: Let the induction hypothesis be true for the n-th step of the algorithm 1. For the $(n+1)$-th step, let the new $\boldsymbol{F}_{S_0}$ and $\boldsymbol{F}_S$ be $\boldsymbol{F}'_{S_0}$ and $\boldsymbol{F}'_S$ respectively. Consider the following two cases:

1. Let $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ be sufficient at line 12. In this case, $\boldsymbol{F}'_{S_0} = \boldsymbol{F}_{S_0}$. So, the induction hypothesis holds.

2. Let $\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$ be not sufficient at line 12.
   Claim: $\boldsymbol{F}_{S_0} \cap p' = \{\}$
   Let the above claim be false.
   $\implies \boldsymbol{F}_{S_0} \cap p' \neq \{\}$
   $\implies \mathcal{F} \setminus (\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}) \subset p'$
   $\implies \sum_{\mathcal{F}n_i \in \mathcal{F} \setminus (\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1})} P_{ub} < \lfloor \frac{\Lambda(\mathcal{A})}{P_{max}} \rfloor$ [From (6)]
   $\implies (\boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1})$ is sufficient. (From Lemma 4)
   $\implies$ Contradiction.
   So, $\boldsymbol{F}_{S_1} \cap p' = \{\}$. In this step, $\boldsymbol{F}'_{S_0} = \boldsymbol{F}_{S_0} \cup \boldsymbol{F}_{S_1}$. Also, from induction hypothesis, $\boldsymbol{F}_{S_0} \cap p' = \{\}$. Therefore, the induction hypothesis holds, i.e., $\boldsymbol{F}'_{S_0} \cap p' = \{\}$.

□

C. Experimental setup

For evaluation we use convolutional networks trained on two popular datasets - MNIST (LeCun et al., 1989) CIFAR-10 (Krizhevsky, 2009) shown in Table 1. The networks are trained with standard training and three state-of-the-art robust training methodologies - adversarial training (PGD training) (Madry et al., 2018), certified robust training (CROWN-IBP) (Zhang et al., 2020) and a combination of both adversarial and certified training (COLT) (Balunovic and Vechev, 2020). For experiments, we use pre-trained publically available networks - the standard and PGD networks are from the ERAN project (Singh et al., 2019c), COLT and CROWN-IBP networks taken from COLT (Balunovic and Vechev, 2020) and CROWN-IBP repository (Zhang et al., 2020) respectively. Similar to most of the existing works on DNN verification (Singh et al., 2019c; Zhang et al., 2018), we use L_∞-based local robustness properties (Carlini and Wagner, 2017). Here, the input region ϕ contains all images obtained by perturbing the intensity of each pixel in the input image independently within a bound ϵ. ψ specifies a region where the network output for the correct class is higher than all other classes. We use $\epsilon_{train} = 0.1$ and $\epsilon_{train} = 8/255$ for training all robustly trained MNIST and CIFAR-10 networks respectively. Unless specified otherwise, the proofs are generated by running the state-of-the-art incomplete verifier α-Crown (Xu et al., 2021). We run all experiments on a 16-core 12th-gen i7 machine with 16 GB of RAM.

D. Additional Experiments

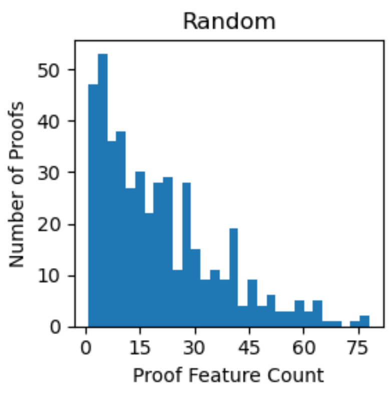
D.1. Detailed results of ProFIt efficacy analysis

Table 2. ProFIt Efficacy Analysis

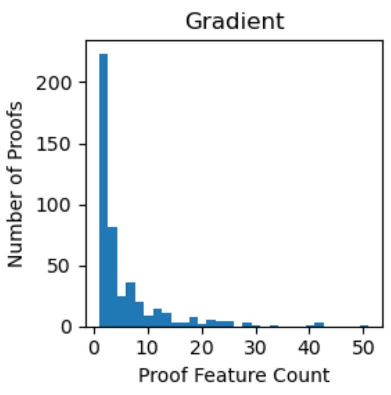
Dataset	Training Method	Input Region (ϕ) eps (ϵ)	No. of proved properties	Original Feature Count	Mean Proof Feature Count			No. of proofs with ≤ 5 proof features (ProFIt)	No. of proofs with ≤ 10 proof features (ProFIt)
					Random	Gradient	ProFIt		
MNIST	Standard	0.02	459	100	20.31	5.25	1.96	449	457
	PGD Trained	0.02	415	1000	93.29	13.73	6.02	315	364
	COLT	0.02	480	100	14.45	5.43	3.46	401	461
	CROWN-IBP	0.02	482	100	9.51	6.73	6.16	240	401
MNIST	PGD Trained	0.1	191	1000	162.39	35.79	3.29	131	149
	COLT	0.1	281	100	29.57	12.22	3.16	240	271
	CROWN-IBP	0.1	473	100	10.09	7.36	6.23	232	384
CIFAR-10	Standard	$0.2/255$	277	100	30.36	18.28	11.12	127	173
	PGD Trained	$0.2/255$	298	100	31.22	16.58	9.74	173	210
	COLT	$0.2/255$	267	250	30.10	18.13	9.03	170	204
	CROWN-IBP	$0.2/255$	265	256	7.96	7.49	5.30	172	221
CIFAR-10	PGD Trained	$2/255$	173	100	39.57	24.46	6.19	122	144
	COLT	$2/255$	229	250	34.64	23.25	7.76	146	181
	CROWN-IBP	$2/255$	206	256	9.41	9.21	5.10	140	176

D.2. Distribution plots of the size of the extracted sufficient proof feature set

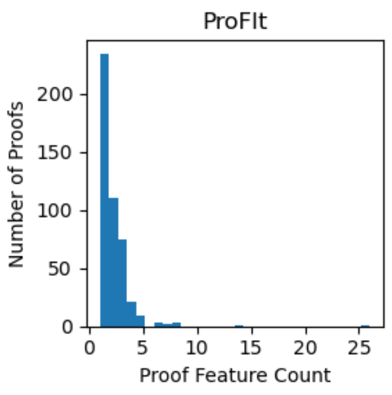
In this section, we show plots for the distribution of the sufficient proof feature set size extracted by ProFIt and two baseline methods based on random and gradient-based priority heuristics. In the following histograms, the x-axis represents the size of the extracted proof feature set and the y-axis represents the number of local robustness properties. The results show that ProFIt consistently outperforms both the baselines and extracts sufficient proof feature sets that are smaller in size. All the plots in this section are generated on 500 local robustness properties with α-Crown verifier.



(a) Random

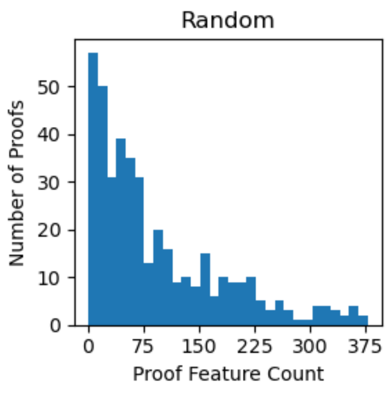


(b) Gradient

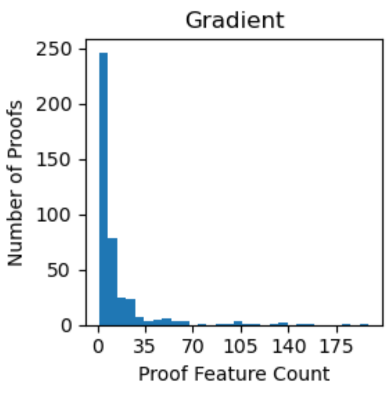


(c) ProFIt

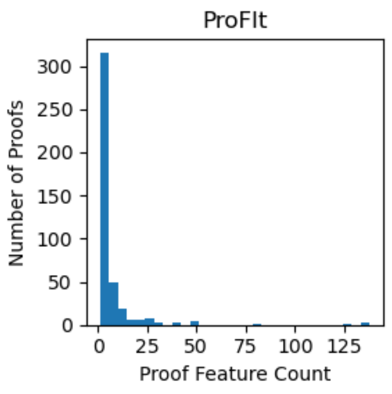
Figure 3. Distribution of the extracted proof feature set size - Standard MNIST network & $\epsilon = 0.02$.



(a) Random

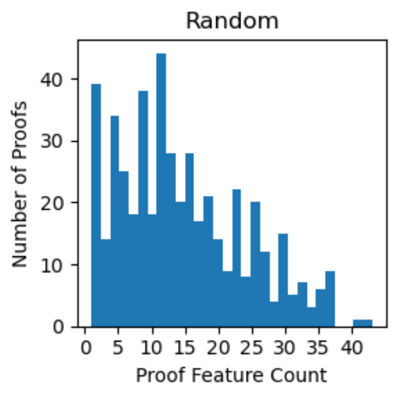


(b) Gradient

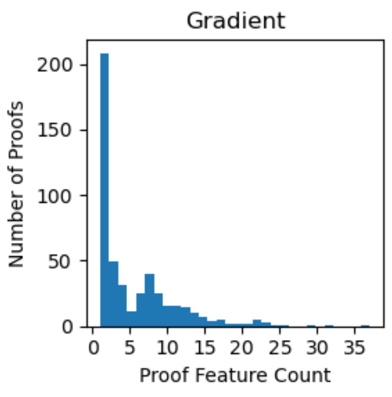


(c) ProFIt

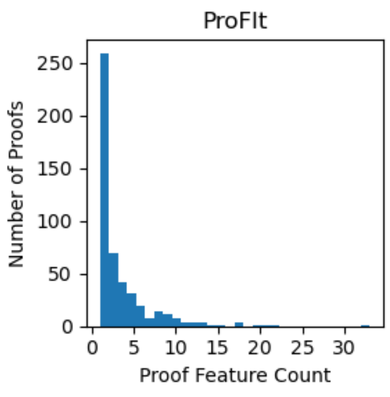
Figure 4. Distribution of the extracted proof feature set size - PGD MNIST network & $\epsilon = 0.02$.



(a) Random

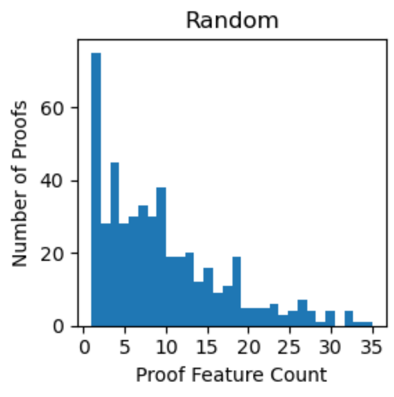


(b) Gradient

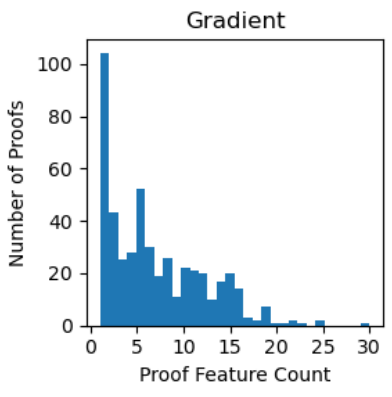


(c) ProFIt

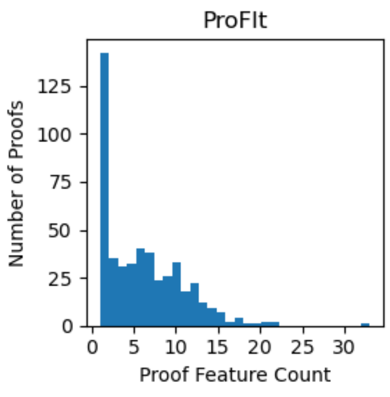
Figure 5. Distribution of the extracted proof feature set size - Colt MNIST network & $\epsilon = 0.02$.



(a) Random

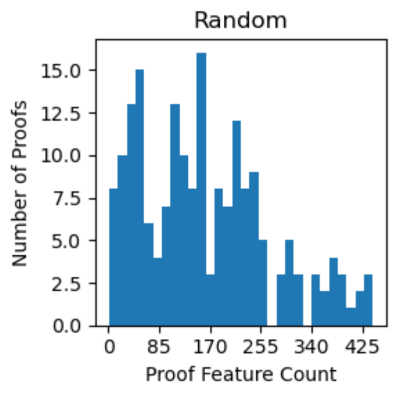


(b) Gradient

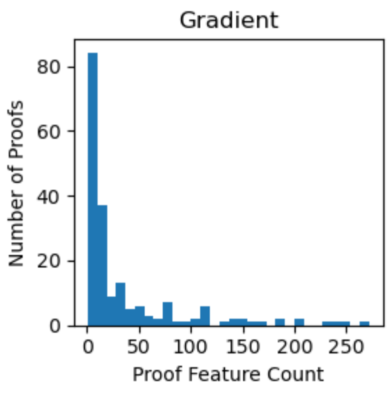


(c) ProFIt

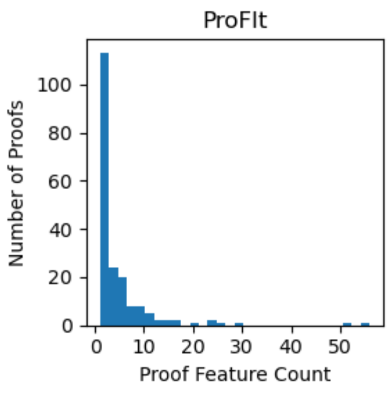
Figure 6. Distribution of the extracted proof feature set size - Crown-IBP MNIST network & $\epsilon = 0.02$.



(a) Random

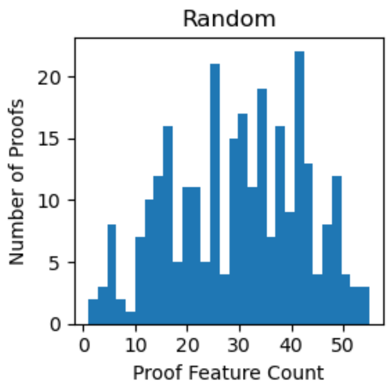


(b) Gradient

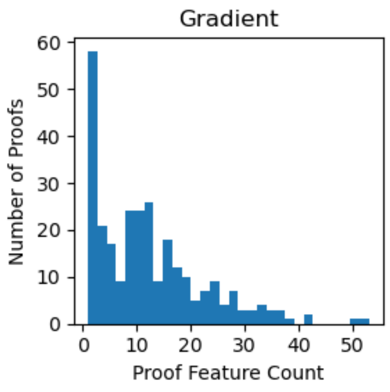


(c) ProFIt

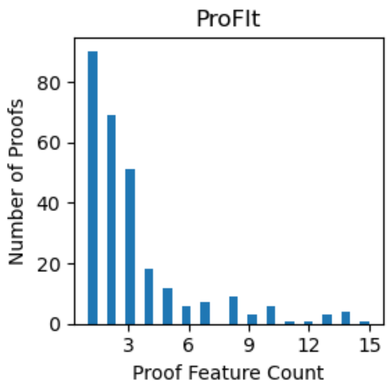
Figure 7. Distribution of the extracted proof feature set size - PGD MNIST network & $\epsilon = 0.1$.



(a) Random

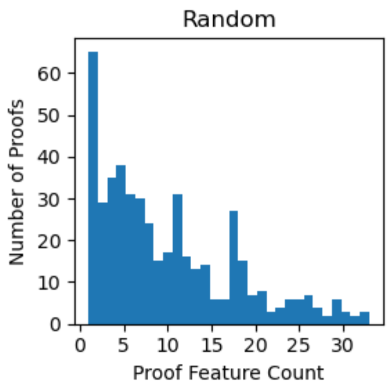


(b) Gradient

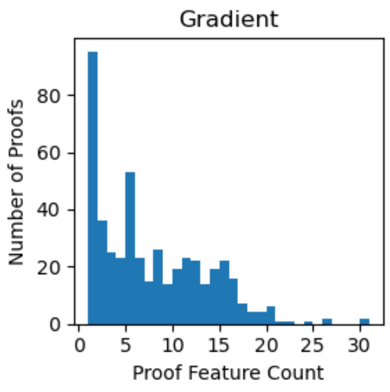


(c) ProFIt

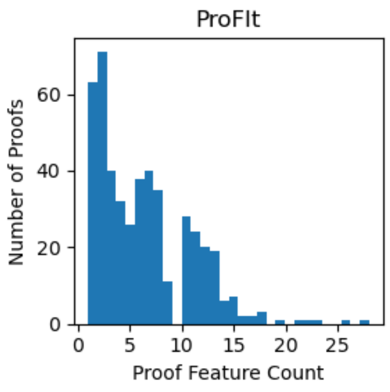
Figure 8. Distribution of the extracted proof feature set size - Colt MNIST network & $\epsilon = 0.1$.



(a) Random

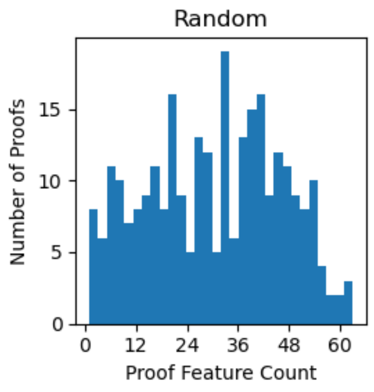


(b) Gradient

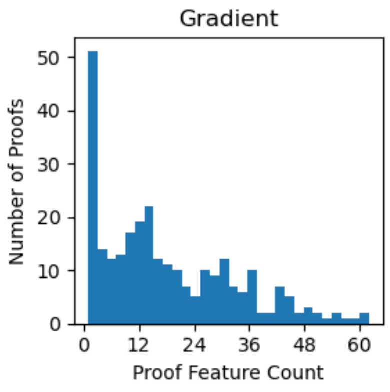


(c) ProFIt

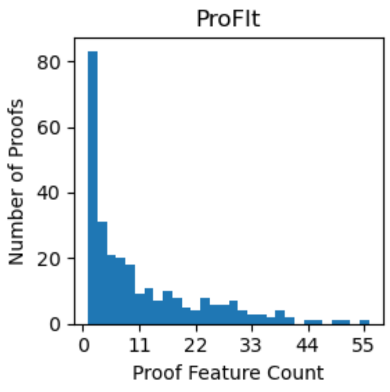
Figure 9. Distribution of the extracted proof feature set size - Crown-IBP MNIST network & $\epsilon = 0.1$.



(a) Random

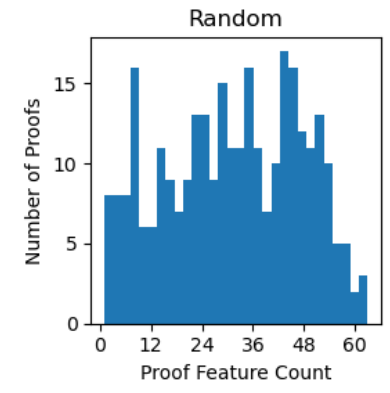


(b) Gradient

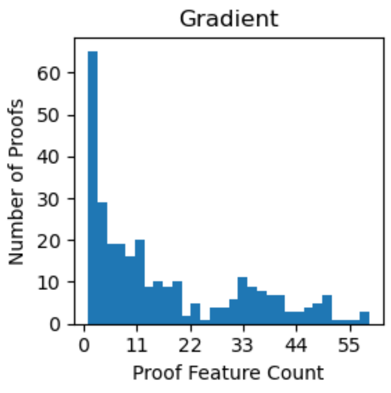


(c) ProFIt

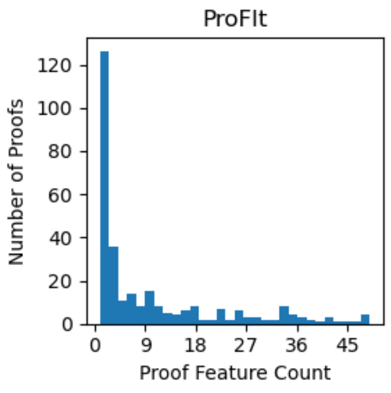
Figure 10. Distribution of the extracted proof feature set size - Standard Cifar-10 network & $\epsilon = 0.2/255$.



(a) Random

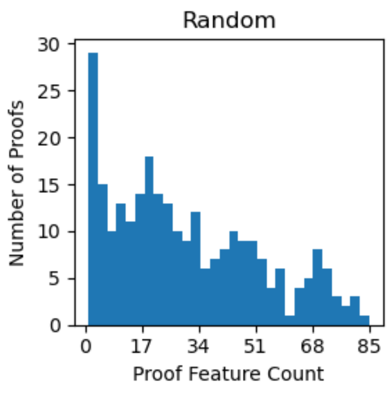


(b) Gradient

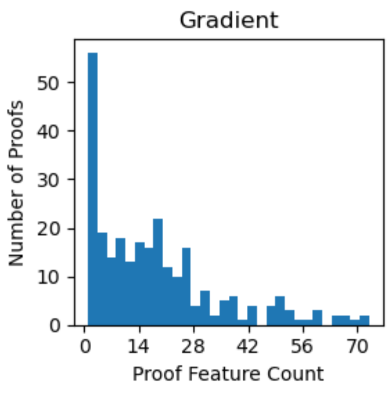


(c) ProFIt

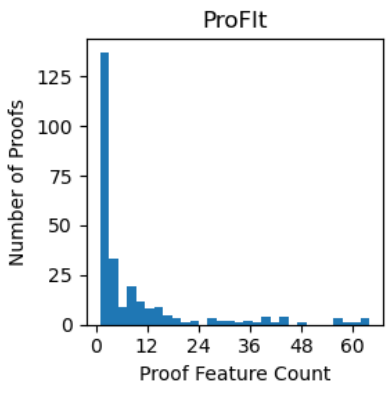
Figure 11. Distribution of the extracted proof feature set size - PGD Cifar-10 network & $\epsilon = 0.2/255$.



(a) Random

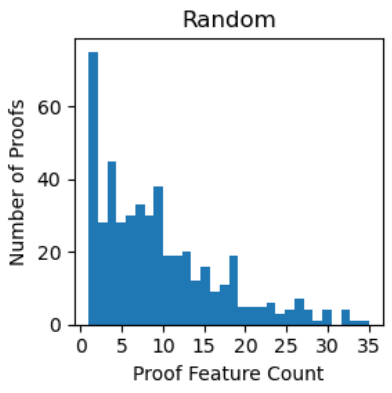


(b) Gradient

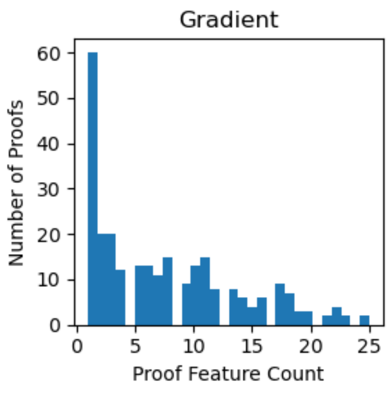


(c) ProFIt

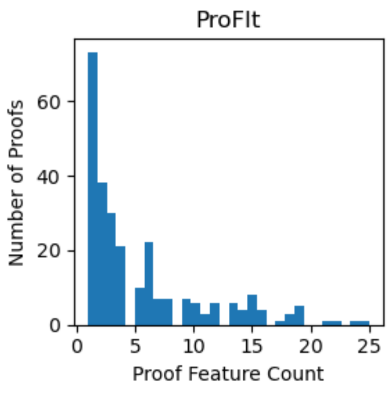
Figure 12. Distribution of the extracted proof feature set size - Colt Cifar-10 network & $\epsilon = 0.2/255$.



(a) Random

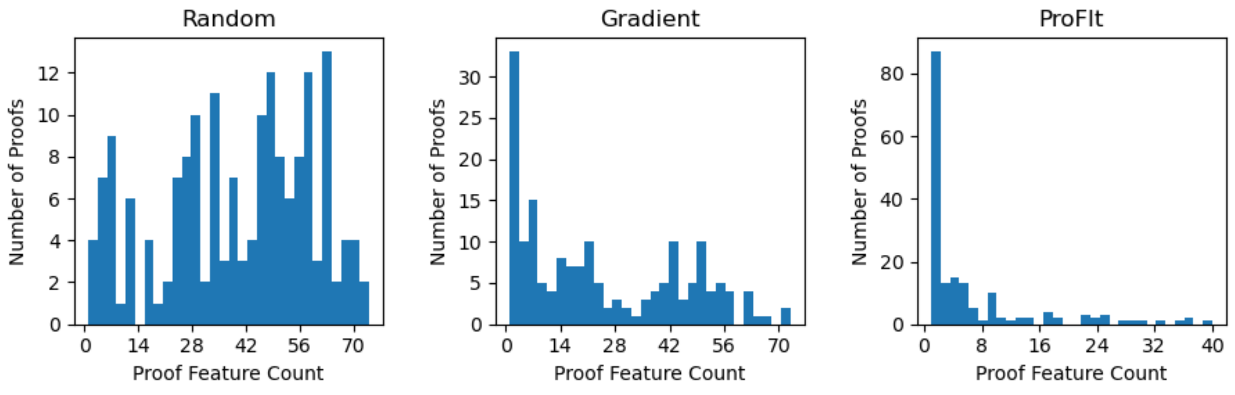


(b) Gradient



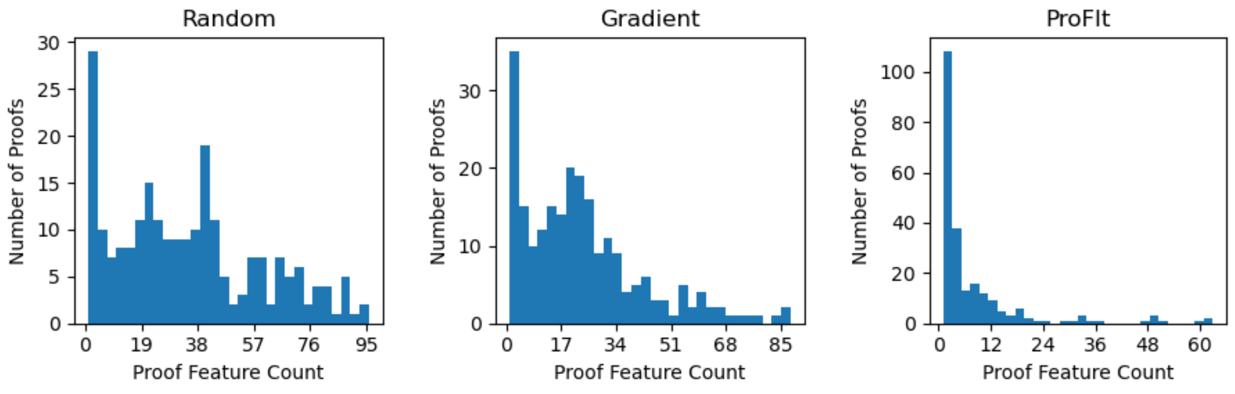
(c) ProFIt

Figure 13. Distribution of the extracted proof feature set size - Crown-IBP Cifar-10 network & $\epsilon = 0.2/255$.



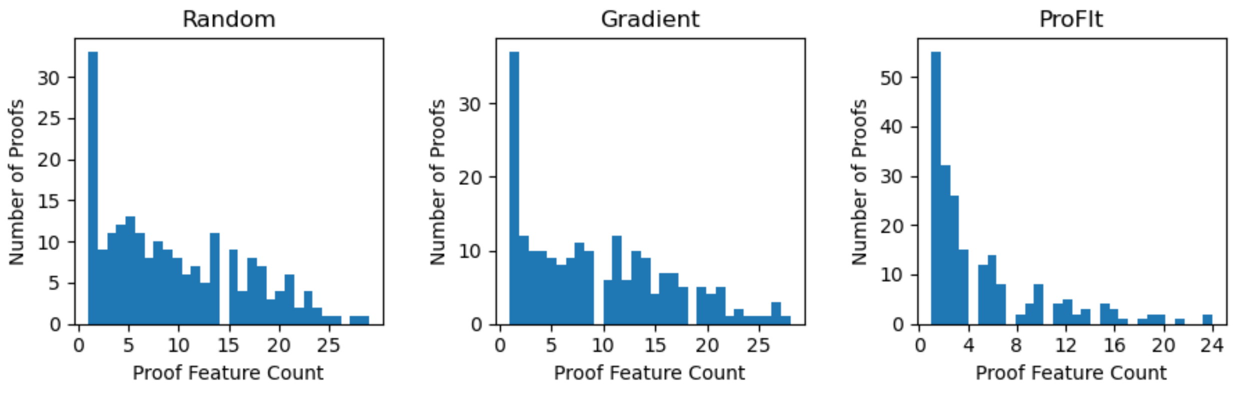
(a) Random (b) Gradient (c) ProFIt

Figure 14. Distribution of the extracted proof feature set size - PGD Cifar-10 network & $\epsilon = 2/255$.



(a) Random (b) Gradient (c) ProFIt

Figure 15. Distribution of the extracted proof feature set size - Colt Cifar-10 network & $\epsilon = 2/255$.

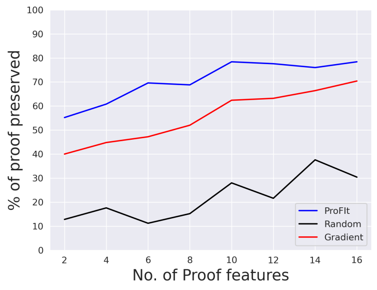


(a) Random (b) Gradient (c) ProFIt

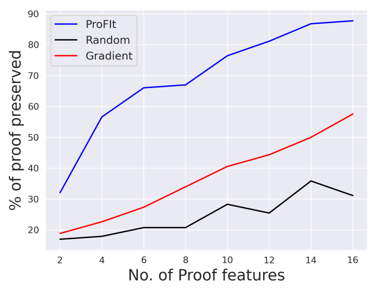
Figure 16. Distribution of the extracted proof feature set size - Crown-IBP Cifar-10 network & $\epsilon = 2/255$.

D.3. Additional plots for priority order evaluation

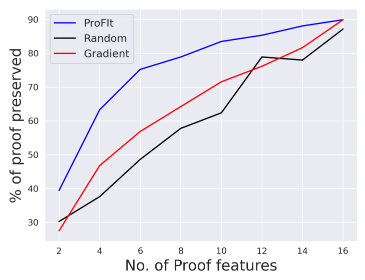
In this section, we evaluate the efficacy of the priority ordering of proof features defined in Eq. 2 against the random and gradient-based priority ordering on Cifar-10 networks and the standard MNIST network. We use $\epsilon = 0.2/255$ for all Cifar-10 networks and $\epsilon = 0.02$ for the standard MNIST network. We show that the priority ordering used by ProFIt preserves a higher % of proofs while better approximating the original verifier output i.e. achieving a lower relative change compared to both the baselines. All the plots in this section generated on 500 local properties with α-Crown verifier.



(a) PGD Network

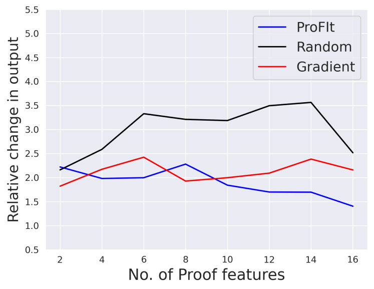


(b) Colt Network

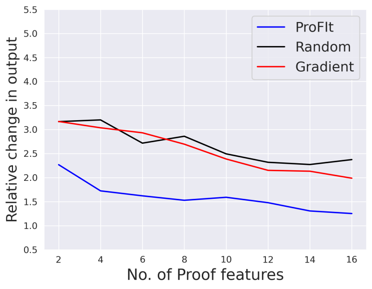


(c) Crown-IBP Network

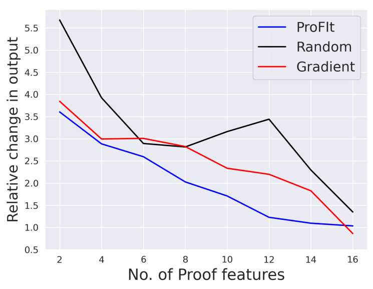
Figure 17. Percentages of proofs preserved by different heuristics on robust CIFAR-10 networks.



(a) PGD Network

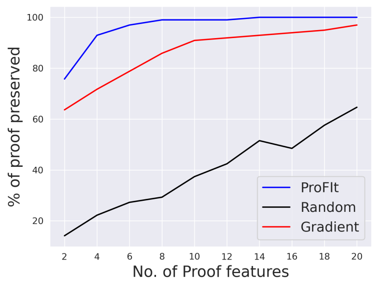


(b) Colt Network

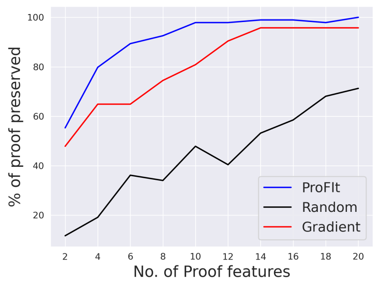


(c) Crown-IBP Network

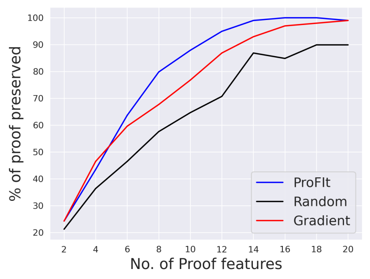
Figure 18. Relative change in verifier output with different heuristics on robust CIFAR-10 networks.



(a) PGD Network

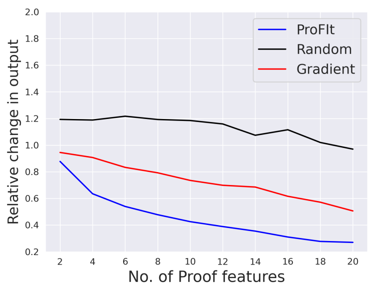


(b) Colt Network

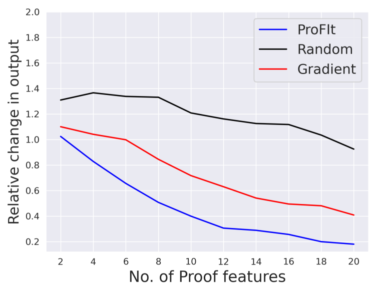


(c) Crown-IBP Network

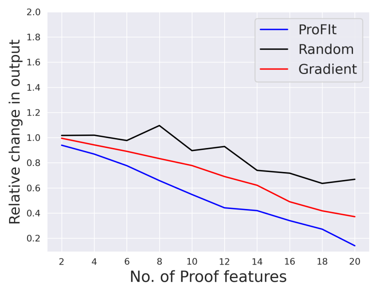
Figure 19. Percentages of proofs preserved by different priority heuristics on robust MNIST networks.



(a) PGD Network

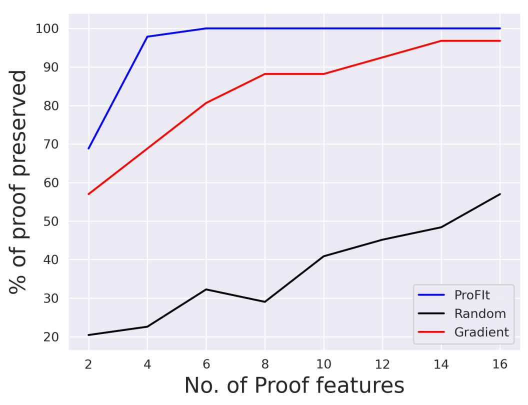


(b) Colt Network

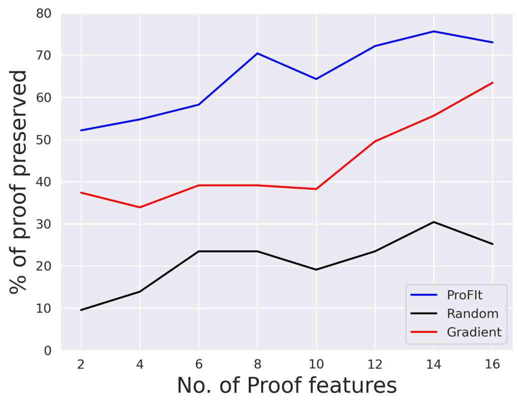


(c) Crown-IBP Network

Figure 20. Relative change in verifier output with different heuristics on robust MNIST networks.

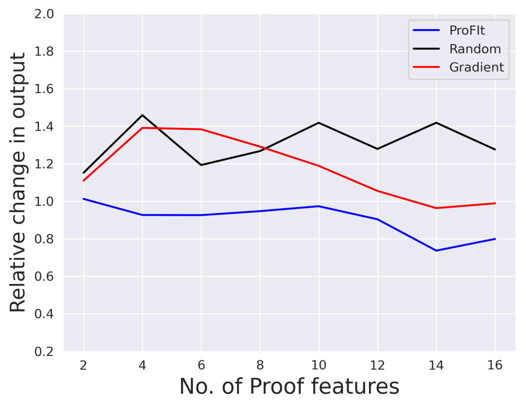


(a) Standard MNIST Network

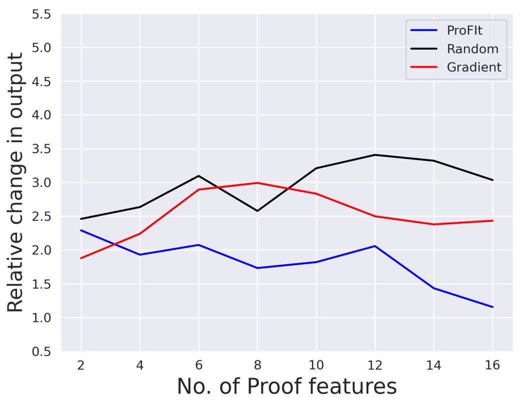


(b) Standard CIFAR-10 Network

Figure 21. Percentages of proofs preserved by different heuristics on standard networks.



(a) Standard MNIST Network



(b) Standard CIFAR-10 Network

Figure 22. Relative change in verifier output with different heuristics on standard networks.

D.4. Qualitative evaluation of the priority ordering of the proof features

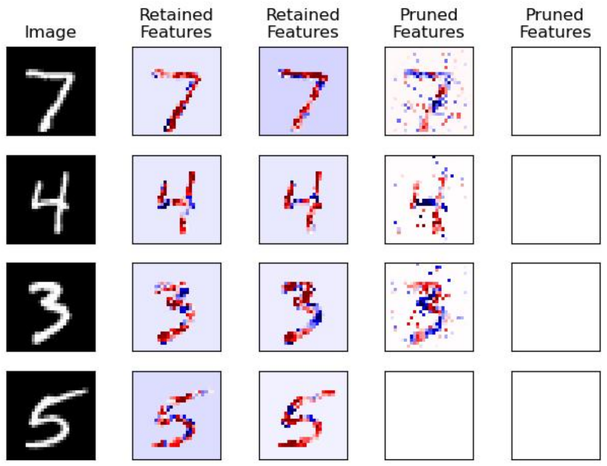
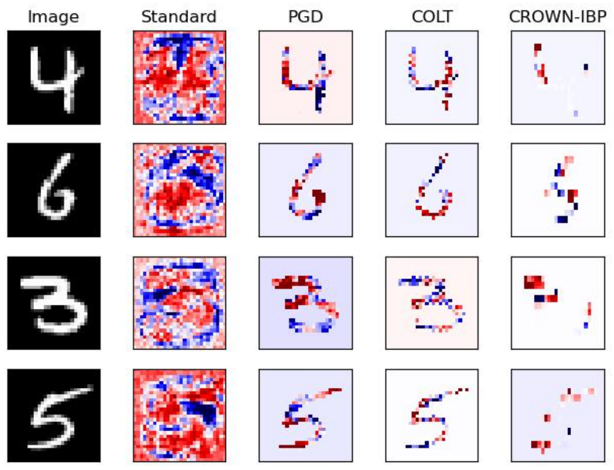
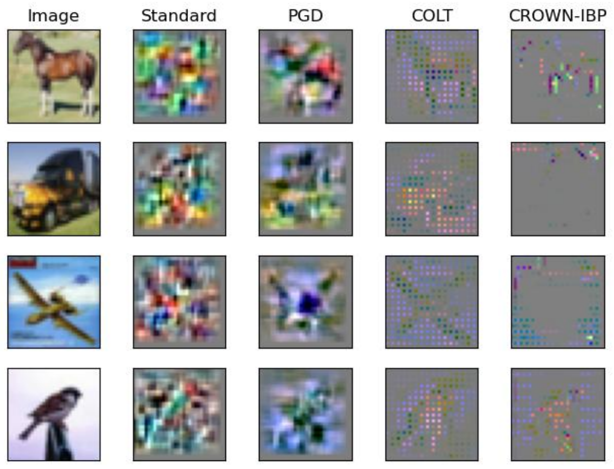


Figure 23. Comparing gradients of the top proof features retained by the ProFIt algorithm to pruned proof features with a low priority. The columns are sorted in decreasing order of priority with the right-most proof feature having the lowest priority. As expected, proof features with low priority are noisier (as in column 4). The proof features which are further down in the priority order do not give any relevant information about the input (column 5 - completely white). This shows that the proposed algorithm extracts proof features that are important to the proof while removing insignificant and uninformative proof features. The shown gradients are computed on COLT-trained MNIST network.

D.5. Additional plots for the top proof feature visualization



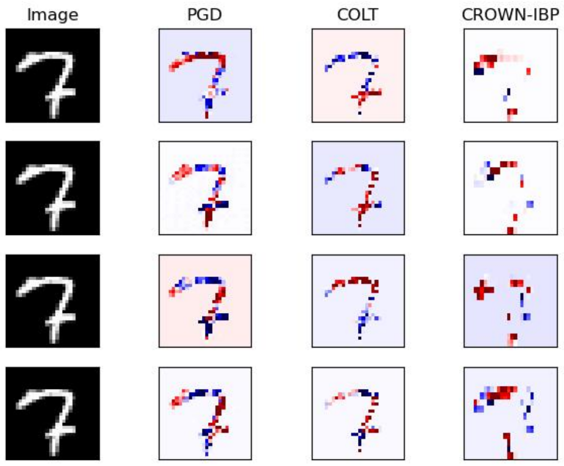
(a) Gradient maps generated on MNIST networks



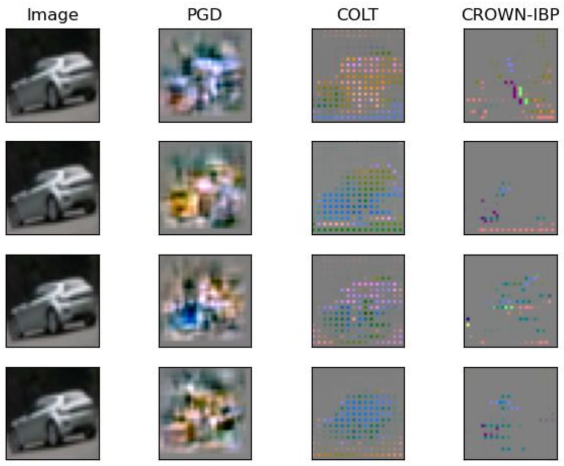
(b) Gradient maps generated on CIFAR-10 networks

Figure 24. Additional plots for the top proof feature visualization (in addition to Fig. 2) - Visualization of gradient map of top proof feature (having highest priority) generated for networks trained with different training methods. It is evident that the top proof feature corresponding to the standard network highlights both relevant and spurious input features. In contrast, the top proof feature of the provably robust network does filter out the spurious input features, but it comes at the expanse of some important input features. The top proof features of the networks trained with PGD filter out more spurious features as compared to standard networks. Finally, the top proof features of the networks trained with COLT filter out the spurious input features and also correctly highlight the relevant input features.

D.6. Visualization of multiple proof features from the extracted proof feature set

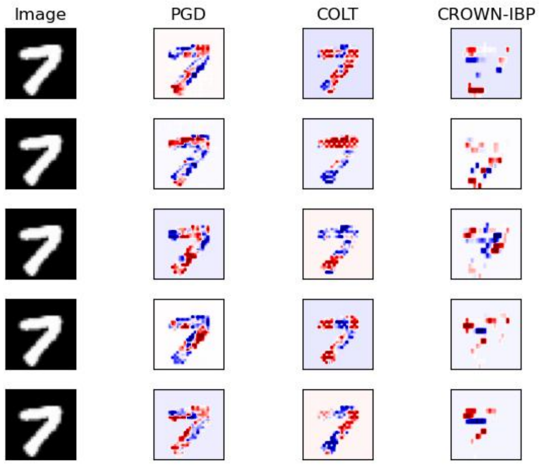


(a) Gradient maps generated on MNIST networks

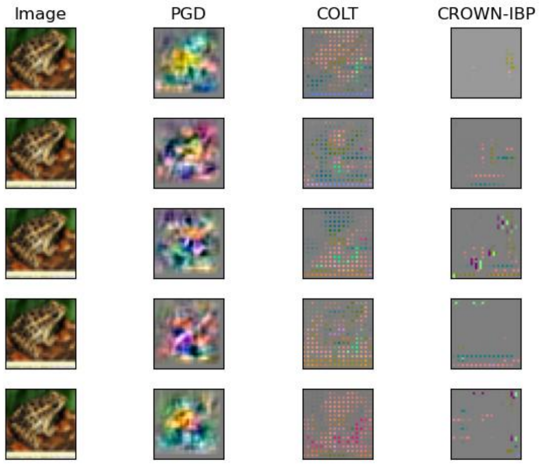


(b) Gradient maps generated on CIFAR-10 networks

Figure 25. Visualization of gradient maps of top-4 proof features (having highest priority) extracted for networks trained with different robust training methods. The gradient maps of the proof features are presented in decreasing order of priority with the top row showing the gradient map corresponding to the top proof feature of each network.



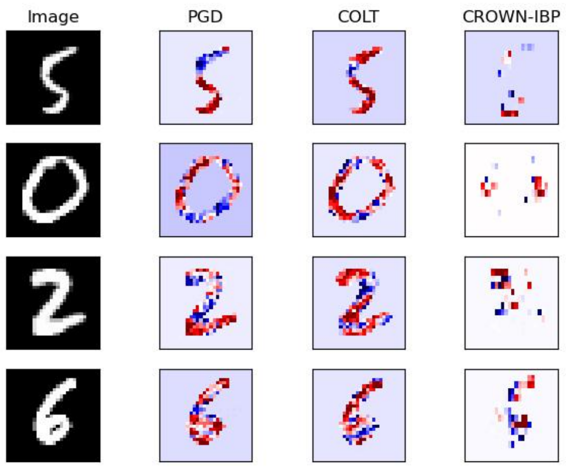
(a) Gradient maps generated on MNIST networks



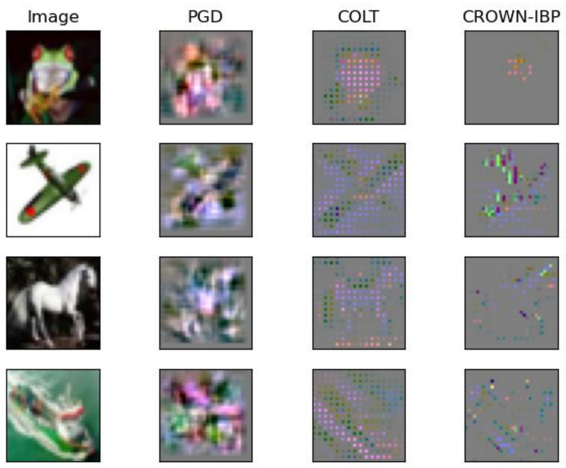
(b) Gradient maps generated on CIFAR-10 networks

Figure 26. Visualization of gradient maps of top-5 proof features (having highest priority) extracted for networks trained with different robust training methods. The gradient maps of the proof features are presented in decreasing order of priority with the top row showing the gradient map corresponding to the top proof feature of each network.

D.7. Visualization of the top proof feature for higher ϵ values



(a) Gradient maps generated on MNIST networks



(b) Gradient maps generated on CIFAR-10 networks

Figure 27. Visualization of gradient map of top proof feature (having highest priority) generated for networks trained with different robust training methods. For these networks, we define local properties with higher ϵ values. For MNIST networks and CIFAR-10 networks, we take $\epsilon = 0.1$ and $\epsilon = 2/255$ respectively.

D.8. Comparing proofs on Networks with same architecture

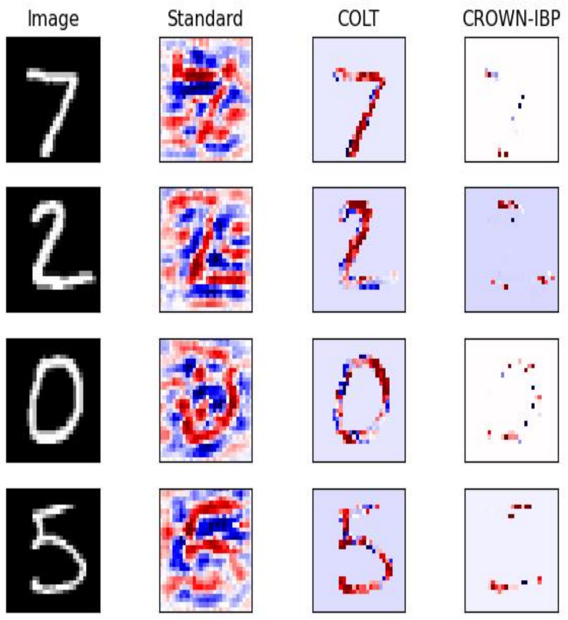


Figure 28. Gradient maps generated on MNIST networks trained with different training methods (Standard, COLT, CROWN-IBP) with the same architecture. The gradient maps show that observations in Section 4.2 and in Figure 2 of the paper also hold on different networks with the same architecture.

D.9. Plots for sensitivity analysis w.r.t ϵ_{train}

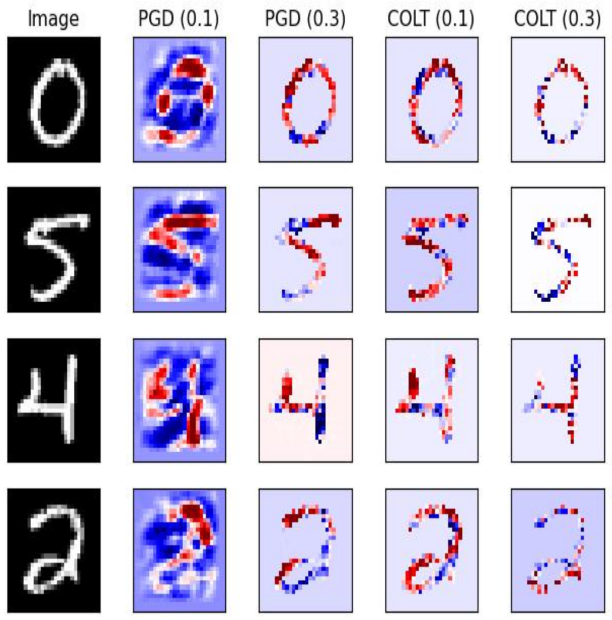


Figure 29. Plots for visualizing gradients of the top proof feature for PGD and COLT networks trained using different values of $\epsilon_{train} \in \{0.1, 0.3\}$ The gradient map corresponding to the networks trained with the higher value of ϵ_{train} filter out more input features than the ones trained with smaller ϵ_{train} value.

E. Ablation Studies

E.1. Ablation study with different verifiers

As described earlier, the extracted proof features are specific to the proof generated by a particular verifier. In this section, we examine whether the proof features corresponding to different proofs generated by different verifiers are the same. In Table 3, we compare the top proof features (ordered by the priority order described in Eq. 2) of the proofs generated by the DeepZ (Singh et al., 2018) verifier and α-Crown verifier. We observed that more than 97% of the cases the top feature remains the same. In this case, for different verifiers though the proof features $\mathcal{F}_{n_i} = [l_{n_i}, u_{n_i}]$ change, their relative priority ordering computed by Eq. 2 remains the same. For this experiment, we use 500 local robustness properties for each network and each ϵ value. Note, while comparing proof features from two different verifiers we only consider those properties that can be proved by both the verifiers.

Table 3. Comparing extracted proof features of DeepZ & α-Crown

Dataset	Training Method	Input Region (ϕ) eps (ϵ)	% properties proved by DeepZ	% properties proved by α-Crown	% proofs with the same top feature	% proofs with the same top-5 feature	% proofs with the same top-10 feature
MNIST	Standard	0.02	90.0 %	91.8 %	99.8 %	98.4 %	98.3 %
	PGD Trained	0.02	82.0 %	83.0 %	99.75 %	99.0 %	98.0 %
	COLT	0.02	95.4 %	96 %	99.50 %	98.95 %	96.25 %
	CROWN-IBP	0.02	96.4 %	96.4 %	99.8 %	99.0 %	95.9 %
MNIST	PGD Trained	0.1	32.6 %	38.2 %	99.38 %	95.7 %	91.41 %
	COLT	0.1	43.0 %	56.2 %	98.6 %	93.95 %	87.90 %
	CROWN-IBP	0.1	89.4 %	94.6 %	97.0 %	88.3 %	80.26 %
CIFAR-10	Standard	0.2/255	51.0 %	58.0 %	99.5 %	98.3 %	98.0 %
	PGD Trained	0.2/255	47.0 %	62.5 %	99.7 %	98.5 %	97.8 %
	COLT	0.2/255	53.0 %	53.0 %	100.0 %	99.5 %	98.2 %
	CROWN-IBP	0.2/255	54.5 %	54.5 %	100.0 %	98.90 %	97.8 %
CIFAR-10	PGD Trained	2/255	26.5 %	32.5 %	99.7 %	95.45 %	92.45 %
	COLT	2/255	45.5 %	46.0 %	99.8 %	95.9 %	97.3 %
	CROWN-IBP	2/255	37.5 %	38.0 %	99.6 %	97.92 %	95.89 %

Next, in Table 4 we compare the top proof feature (having the highest priority) corresponding to the proofs generated by DeepZ, Crown (Zhang et al., 2018), α-Crown and state-of-the-art complete verifier α, β-Crown (Wang et al., 2021b) on the property. We use PGD and Colt MNIST network for this experiment. We evaluate 200 local robustness properties defined with $\epsilon = 0.02$. We observed that for these two networks, more than 99% of the cases the top feature remains the same.

Table 4. % cases different verifiers have the same top proof feature

Verifier	DeepZ	Crown	α-Crown	α, β-Crown
DeepZ	100.0 %	99.55 %	99.75 %	99.50 %
Crown	99.55 %	100.0 %	99.80 %	99.60 %
α-Crown	99.75 %	99.80 %	100.0 %	99.80 %
α, β-Crown	99.50 %	99.60 %	99.80 %	100.0 %

(a) PGD MNIST Network

Verifier	DeepZ	Crown	α-Crown	α, β-Crown
DeepZ	100.0 %	99.50 %	99.50 %	99.40 %
Crown	99.50 %	100.0 %	100.0 %	99.70 %
α-Crown	99.50 %	100.0 %	100.0 %	99.70 %
α, β-Crown	99.40 %	99.70 %	99.70 %	100.0 %

(b) Colt MNIST Network

E.2. Ablation study with Training parameters

Next, we compare proofs generated on networks with the same architecture trained with different training methods to validate that the underlying network architecture does not play any role in the conclusions presented in Section 4.2. We also analyze the sensitivity of the extracted proof features to the training parameter ϵ_{train} that is used to define the L_∞ region during training (Appendix D.9). We observe that networks trained with higher ϵ_{train} are more robust and the top-proof feature filters out more input features that align with the observations in Section 4.2.

F. Related Work

DNN interpretability. There has been an extensive effort to develop interpretability tools for investigating the internal workings of DNNs. These include feature attribution techniques like saliency maps (Sundararajan et al., 2017; Smilkov et al., 2017), using surrogate models to interpret local decision boundaries (Ribeiro et al., 2016), finding influential (Koh and Liang, 2017), prototypical (Kim et al., 2016), or counterfactual inputs (Goyal et al., 2019), training sparse decision layers (Wong et al., 2021), utilizing robustness analysis (Hsieh et al., 2021). Most of these interpretability tools focus on generating local explanations that investigate how DNNs work on individual inputs. Another line of work, rather than explaining individual inputs, tries to identify specific concepts associated with a particular neuron (Simonyan et al., 2014; Bau et al., 2020). However, to the best of our knowledge, there is no existing work that allows us to interpret DNN robustness proofs.

DNN verification. Unlike DNN interpretability methods, prior works in DNN verification focus on formally proving whether a DNN satisfies desirable properties like robustness (Singh et al., 2019c; Wang et al., 2021b), fairness (Mazzucato and Urban, 2021), etc. The DNN verifiers are broadly categorized into three main categories - (i) sound but incomplete verifiers which may not always prove property even if it holds (Gehr et al., 2018; Singh et al., 2018; 2019b;a; Zhang et al., 2018; Xu et al., 2020; Salman et al., 2019), (ii) complete verifiers that can always prove the property if it holds (Wang et al., 2018; Gehr et al., 2018; Bunel et al., 2020a;b; Bak et al., 2020; Ehlers, 2017; Ferrari et al., 2022; Fromherz et al., 2021; Wang et al., 2021a; Palma et al., 2021; Anderson et al., 2020; Zhang et al., 2022) and (iii) verifiers with probabilistic guarantees (Cohen et al., 2019).

Robustness and interpretability. Existing works (Madry et al., 2018; Balunovic and Vechev, 2020; Zhang et al., 2020) in developing robust training methods for neural networks provide a framework to produce networks that are inherently immune to adversarial perturbations in input. Recent works (Tsipras et al., 2019; Zhang et al., 2019) also show that there may be a robustness-accuracy tradeoff that prevents highly robust models achieve high accuracy. Further, in (Tsipras et al., 2019) authors show that networks trained with adversarial training methods learn fundamentally different input feature representations than standard networks where the adversarially trained networks capture more human-aligned data characteristics.