Robustness Assessment of a Runway Object Classifier for Safe Aircraft Taxiing

Yizhak Elboher*§, Raya Elsaleh*§ Omri Isac*§, Mélanie Ducoffe†, Audrey Galametz†, Guillaume Povéda†, Ryma Boumazouza†, Noémie Cohen† and Guy Katz*
*The Hebrew University of Jerusalem †Airbus Central Research & Technology, AI Research

2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC) | 979-8-3503-4961-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/DASC62030.2024.10748680

***Abstract*—As deep neural networks (DNNs) are becoming the prominent solution for many computational problems, the aviation industry seeks to explore their potential in alleviating pilot workload and improving operational safety. However, the use of DNNs in these types of safety-critical applications requires a thorough certification process. This need could be partially addressed through formal verification, which provides rigorous assurances — e.g., by proving the absence of certain mispredictions. In this case-study paper, we demonstrate this process on an image-classifier DNN currently under development at Airbus, which is intended for use during the aircraft taxiing phase. We use formal methods to assess this DNN's robustness to three common image perturbation types: *noise*, *brightness* and *contrast*, and some of their combinations. This process entails multiple invocations of the underlying verifier, which might be computationally expensive; and we therefore propose a method that leverages the monotonicity of these robustness properties, as well as the results of past verification queries, in order to reduce the overall number of verification queries required by nearly 60%. Our results indicate the level of robustness achieved by the DNN classifier under study, and indicate that it is considerably more vulnerable to noise than to brightness or contrast perturbations.**

I. Introduction

In recent years, deep neural networks (DNNs) have been revolutionizing computer science, advancing the state of the art in many domains [17] — including natural language processing, computer vision, and many others. In the aviation domain, aircraft manufacturers are now exploring how deep-learning-based technologies could decrease the cognitive load on pilots, while increasing the safety and operational efficiency of, e.g., airports. In particular, these technologies could prove useful during the aircraft taxi phase, which often creates an increased cognitive load on pilots who have to simultaneously manage the flight plan, the aircraft itself, and any objects on the tarmac.

Despite their success, DNNs are known to be prone to various errors. Notable among these are *adversarial inputs* [6], which are slightly perturbed inputs that lead to incorrect and potentially unsafe DNN outputs. While there exist many techniques for efficiently finding adversarial inputs, it is unclear how to certify that no such examples exist. However, such a certification process will be required to allow the integration of DNNs into safety-critical industrial systems, e.g., in aviation.

Aviation authorities involved in managing aircraft certification, such as the European Union Aviation Safety Agency (EASA), have recently published the key elements required for certifying DNN models to be used in aviation.[1] There, EASA particularly emphasizes that DNN verification solutions, to be applied during the learning and system integration phases, will likely constitute a *means of compliance* with regulatory requirements.[2] EASA points out, however, that the current scalability and the expressiveness of DNN verification techniques is limited.

Typically, DNN formal verification tools seek to prove that, for a given infinite set of inputs, a DNN only produces outputs that fall within a safe subspace of the output space. To date, these tools have been predominantly applied in assessing the robustness of DNN predictions against specific types of local input perturbations. Maturing these techniques is thus key in allowing them to meet the bar needed for DNN certification in, e.g., aviation. This point is again stressed in EASA's AI Roadmap,[3] which emphasizes the need for providing more general guarantees of a DNN's stability.

Although DNN verification has been making great strides [1, 7, 8, 9, 12, 13, 14, 15, 16, 19], it has so far been applied only to a limited number of real-world systems. In this case-study paper, we study the applicability and scalability of DNN verification through an object classification use-case, relevant to the aviation domain and of specific interest to Airbus. We explore pertinent vision-oriented perturbations (*noise*, *brightness*, and *contrast*) and use formal verification to quantify their effects on DNN's robustness. As a back-end engine, we use the Marabou DNN verifier [18]. We also demonstrate that the verification process can be optimized by leveraging the monotonicity of the studied perturbations.

Our results indicate that while the DNN is highly sensitive to noise perturbations, it is slightly less vulnerable to contrast and brightness perturbations. This is a reassuring result, as these perturbations are strongly correlated with highly unpredictable operating conditions, especially outdoors. More broadly, our results showcase the usefulness and potential of DNN verification in aviation that could easily be extended to other safety-critical domains.

II. Background

Deep Neural Networks. A deep neural network [5] $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^k$ is comprised of m layers, $L_1, ..., L_m$. Each layer

§Authors contributed equally.

[1]https://www.easa.europa.eu/en/downloads/137631/en

[2]https://www.easa.europa.eu/en/document-library/general-publications/concepts-design-assurance-neural-networks-codann

[3]https://www.easa.europa.eu/en/domains/research-innovation/ai

L_i consists of a set of nodes, S_i. When $\mathcal{N}$ is evaluated, each node in the input layer is assigned an initial value. Then, the value of the j^{th} node in the $2 \leq i < m$ layer, v_j^i, is computed as:

$$v_j^i = f\left(\sum_{l=1}^{|S_{i-1}|} w_{j,l}^{i-1} \cdot v_l^{i-1} + b_j^i\right)$$

where $f : \mathbb{R} \to \mathbb{R}$ is an *activation function* and $w_{j,l}^{i-1}, b_j^i \in \mathbb{R}$ are the respective *weights* and *biases* of $\mathcal{N}$. The most common activation function is the *rectified linear unit (ReLU)*, defined as $\text{ReLU}(x) = max(0, x)$. Finally, neurons in the output layer are assigned values using an affine combination only. The output of the DNN is the values of the nodes in its final layer. An image-classifier $\mathcal{N} : \mathbb{R}^n \to C \subset \mathbb{N}$ assigns each input image x' a class $c \in C$, which describes the main object depicted in x'. For convenience, x' is regarded as both a vector and a matrix, interchangeably. For an example of a DNN and its evaluation, see Appendix B.

DNN Verification. For a DNN $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^k$, input property $P \subset \mathbb{R}^n$ and output property $Q \subset \mathbb{R}^k$, the *DNN verification problem* is to decide whether there exist $x \in P$ and $y \in Q$ such that $\mathcal{N}(x) = y$. If such a pair exists, the verification query $(\mathcal{N}, P, Q)$ is *satisfiable* (SAT), and the pair (x, y) is called a witness; otherwise, it is *unsatisfiable* (UNSAT). Typically, Q encodes an undesired behavior, and so a witness is a *counterexample* that demonstrates an error.

III. Industrial Use-Case: Runway Object Classification

A. Runway Object Classification

In 2020, Airbus concluded its Autonomous Taxi, Take-Off and Landing (ATTOL) project.[4] The objective of ATTOL was to design a fully autonomous controller for the taxi, take-off, approach and landing phases of a commercial aircraft — by leveraging state-of-the-art technology, and in particular deep-learning models used for vision-assisted functions. As part of the project, 400 flights over a period of two years were instrumented to collect video data from aircraft in operation. This unique dataset is currently being used to further mature several vision-based functions within Airbus. Using this dataset, it was observed that the taxi phase of the flight, in particular, could benefit from autonomous support. During this phase, pilots are conducting aircraft operations, while simultaneously dealing with the unpredictable nature of airport management and traffic. Object identification, in particular of potential threats on the runway, could thus support the pilots during this phase. Several object classifiers are being tested for this purpose within Airbus.

In this study, we focus on images of runway objects extracted from taxiing videos — i.e., all objects are observed from an aircraft on the ground. We extract (224×224) pixel images from the original, high-resolution gray-scaled images, centered on a specific runway object. A DNN N_1 is trained on resampled (32×32) images, to improve verification performance. The four considered classes are *Aircraft*, *Vehicle*, *Person*, and *Negative*, extracted where no object is found. N_1 is a feedforward DNN, with roughly 8000 ReLU neurons, and an accuracy of 85.3% on the test dataset (1145/1342 images).[5]

B. Properties of Interest

We seek to verify the *local robustness* of a runway object classifier $\mathcal{N}$; i.e., that small perturbations around a correctly-classified input x' do not cause misclassification, encoded by Q. We specify Q as: $Q_{x'} := C \setminus \mathcal{N}(x')$. We use the input property P to define three perturbation types: noise, brightness, and contrast.

Noise. In this widely studied form of perturbation [2, 11], the perturbed input images are taken from an ϵ-ball around x': $P = B_\epsilon(x')$, where B_ϵ is the ℓ_∞-ϵ-ball around x', and $\epsilon > 0$.

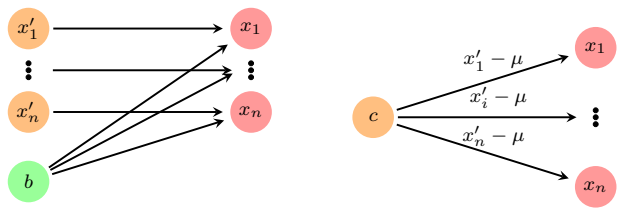
Brightness. A brightness perturbation is caused by shifting all pixels of x' by a constant value b: $bright(x', b) := x' + b \cdot J_n$, where J_n is the all-ones matrix of size $n \times n$. We define $P = bright_\beta(x') := \{bright(x', b) \mid |b| \leq \beta\}$ for some $\beta > 0$, to allow all brightness perturbations of absolute value at most β. See Appendix A for a visual example.

Contrast. A contrast perturbation $con(x', c, \mu)$ is created by scaling all image pixels multiplicatively, rescaling their difference from a mean value $\mu \in [0, 1]$ by a multiplicative constant $c \in \mathbb{R}_{\geq 0}$: $con(x', c, \mu) := \mu \cdot J_n + c \cdot (x' - \mu \cdot J_n)$. We then set $P = con_{\gamma,\mu}(x') := \{con(x', c, \mu) \mid |c - 1| \leq \gamma\}$, to encode all contrast perturbations with value of at most γ, where μ remains constant and $\gamma \in [0, 1]$. See Appendix A for a visual example.

IV. The Formal Verification Process

A. Encoding Brightness and Contrast Perturbations

We now show how to encode the brightness and contrast properties described in Section II into verification queries that assess robustness to noise perturbations over a modified input space. This reduction allows us to use any of the available tools that support such queries as a backend. The encoding is performed by adding a new input layer to the network, as illustrated in Fig. 1.



(a) Brightness. Weights are set to 1, biases to 0.

(b) Contrast. Biases are set to μ.

Fig. 1: Modeling brightness and contrast perturbations by adding an input layer.

[4]https://www.airbus.com/en/newsroom/press-releases/2020-06-airbus-concludes-attol-with-fully-autonomous-flight-tests

[5]These DNNs will not be used as such in Airbus products. More robust models are currently under development, in part supported by analyses such as the one presented here.

Brightness. The new input layer clones the original input layer and adds a single neuron b to represent the brightness perturbations. The weights from the new layer to the following, original input layer are set to 1 so that every variable $x_i \in x$ is assigned $x_i = x'_i + b$. The bounds for the new neuron are set to $b \leq \beta, b \geq -\beta$, whereas inputs x'_i are exactly restricted to the input around which the robustness is being verified. We note that in this case, this single construction allows the verification of the robustness around any input, by selecting appropriate x'_i values. We further note that this construction can be used to simultaneously encode noise and brightness perturbations, by bounding the input neurons x'_i to an ϵ-ball around an input of interest. This gives rise to two-dimensional queries, for any combination of β and ϵ values, which allows modeling a more realistic nature of perturbations.

Contrast. The new input layer contains a single input neuron, c. We treat μ, x' as constants, and set the weights from the new layer in a way that every neuron x_i in that layer is assigned $x_i = (x'_i - \mu) \cdot c + \mu$. Finally, we set the bounds $c \geq 1 - \gamma, c \leq 1 + \gamma$. We note that the contrast perturbation is multiplicative with respect to c, μ, and the input image x'. Since DNN verification algorithms typically only support linear operations, either x' or c should be fixed. Therefore, a separate DNN is constructed for each input image; and there is no immediate way to encode a simultaneous noise perturbation.

B. Incremental Verification Algorithm

For any fixed image x', we seek to solve numerous brightness, noise and contrast robustness queries, with different values of ϵ, β and γ. Since executing these queries is computationally expensive, we exploit the monotonicity of these properties to reduce their number. Let $\beta' < \beta$, $\epsilon' < \epsilon$ and $\gamma' < \gamma$. If there exists an adversarial example for parameters β', ϵ' or γ', it then also constitutes a counterexample for a query with parameters β, ϵ or γ respectively. Conversely, if the network is robust with respect to parameters β, ϵ or γ, then it is also robust to perturbation with parameters β', ϵ' or γ' respectively.

We exploit this property in a binary search algorithm for contrast queries, and in our *incremental verification algorithm* for brightness and noise queries. Intuitively, the algorithm initializes a grid representing all combinations of ϵ, β parameters that need to be verified. The observation above states that for every row and every column, there is at most one transition from `UNSAT` to `SAT`, which is represented by a step graph within the grid. The algorithm then discovers this step graph instead of solving all queries in the grid.

The pseudo-code of the algorithm appears in Algorithm 1. Formally, Algorithm 1 assumes the existence of a verification procedure verify$(\mathcal{N}, x', \beta, \epsilon)$ which verifies the robustness of a network $\mathcal{N}$ to noise perturbations of value at most ϵ and brightness perturbations of value at most β around an image x'. The algorithm is given an input network $\mathcal{N}$, an image x', and two increasingly ordered arrays B, E, containing the values of parameters β and ϵ we intend to check, respectively. Then, the algorithm initializes a grid representing all possible

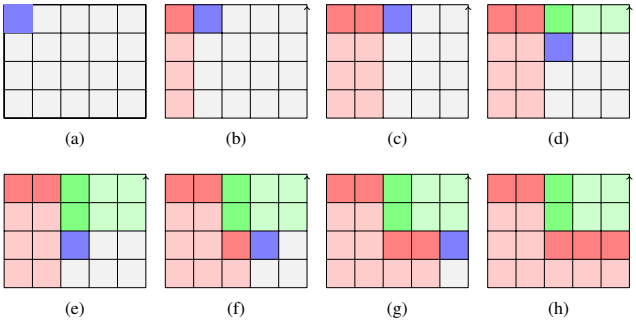


Fig. 2: Example of incremental verification algorithm's run.

combinations of parameters $(\beta, \epsilon) \in B \times E$, and a temporary tuple (b, e) representing the lowest value of β and highest value ϵ and corresponding to the top-left corner of the grid. The algorithm then iteratively calls verify$(\mathcal{N}, x', b, e)$ to populate the grid. If the result is `SAT`, then for all queries with the same ϵ value, and a greater β (all cells to the right of the current cell) the result is `SAT` as well.[6] We then mark the relevant cells with `SAT` and decrement ϵ to the next value. If the result is `UNSAT`, then for all queries with the same β value, and a smaller ϵ (all cells to the bottom of the current cell), the result is `UNSAT` as well. We then mark the relevant cells with `UNSAT` and increment β to the next value. When the current cell reaches the final column or row, a typical binary search algorithm is used to find the remaining results. Note that the algorithm requires only $O(m)$ calls to the verifier while naively solving all queries requires $O(m^2)$ calls, where m is the number of possible values of β or ϵ (the maximal of the two). For contrast queries, the binary search allows using a logarithmic number of invocations of the verifier instead of a linear number.

To support the use of real-world verifiers, we also address cases where the verifier returns a `TIMEOUT` or error value. When these cases occur, we mark the corresponding cell (e, b) with an `UNKNOWN` result, and increment the value of b as if the result was `SAT`. In addition, we use binary search for the remaining values of e, where the value of b is constant. Note that in the presence of `TIMEOUT`, the bound of $O(m)$ calls to the verifier is not guaranteed.

Example. In Fig. 2, the grid represents the options for verification queries of robustness for brightness and noise perturbation, with parameters $(\beta, \epsilon) \in [0.1, 0.2, 0.3, 0.4, 0.5] \times [0.1, 0.2, 0.3, 0.4]$. The purple cell represents the current tuple (β, ϵ). Red marks `UNSAT` queries, green marks `SAT` queries. Rich colors represent a call for the verifier, while pale colors represent a deduction of satisfiability. The algorithm first queries the verifier to verify robustness with parameters $(0.4, 0.1)$, which returns `UNSAT`. Then, the algorithm deduces `UNSAT` for queries with $\beta = 0.1, \epsilon < 0.4$ without calling the verifier again, and queries the verifier to verify robustness with parameters $(0.4, 0.2)$. Since the verifier returns `UNSAT` again, the algorithm deduces `UNSAT` for queries with $\beta = 0.2, \epsilon < 0.4$ and queries the verifier to verify robustness with

[6]Note that this is the case for all queries with greater values of ϵ, β (the top right rectangle), though the values of queries with a greater ϵ value are already decided. A dual argument applies to the `UNSAT` case as well.

Algorithm 1 Incremental verification algorithm

Input: Arrays B, E with values of ϵ, β in increasing order, respectively, a verifier V, a network $\mathcal{N}$ and an image x'.
Output: A grid representing the robustness of $\mathcal{N}$ to brightness and noise perturbations around x', for all values in B, E.

```
b ← 0
e ← length(E) − 1
grid ← 0_{length(B)×length(E)}
while b < length(B) and e ≥ 0 do
    if b = length(B) − 1 then
        Binary search with remaining values of e; b is constant.
    end if
    if e = 0 then
        Binary search with remaining values of b; e is constant.
    end if
    result ← V.verify(N, x', E[e], B[b])
    if result = SAT then
        ∀i ≥ b : grid[i][e] ← SAT
        e ← e − 1
    else if result = UNSAT then
        ∀j ≤ e : grid[b][j] ← UNSAT
        b ← b + 1
    else                                  ▷ Timeout, Memoryout, etc.
        grid[b][e] ← UNKNOWN
        Binary search with remaining values of e; b is constant.
        b ← b + 1
    end if
end while
return grid
```

parameters $(0.4, 0.3)$. This time, the verifier returns SAT, so the algorithm deduces SAT for queries with $\beta > 0.2, \epsilon = 0.4$. The algorithm then queries the verifier to verify robustness with parameters $(0.3, 0.3)$. The rest of the iterations continue similarly.

V. Evaluation

For the 1145 correctly classified test images, we verify N_1's robustness to noise and brightness for parameters $(\epsilon, \beta) \in [0, 0.05, 0.1, 0.15, 0.2] \times [0, 0.1, 0.2, 0.3, 0.4, 0.5]$, and to contrast perturbations with mean pixel value $\mu = 0.2585$ and $\gamma \in [0.1, 0.2, ..., 0.9]$. We make use of the incremental verification algorithm for noise and brightness perturbations. For contrast, we run a binary search algorithm to find the minimal γ parameter for which the query is UNSAT. We use an arbitrary timeout of $22.5K$ seconds per single query, and 80 hours for the overall runtime to analyze a single input point. The results are summarized below and additional details appear in the preprint version of the paper [4].

Fig. 3 shows the percentage of UNSAT queries for noise and brightness perturbations, indicating the absence of counter-examples, of 1097 points for which the analysis has not timed out (the points with timeout analysis were not included).

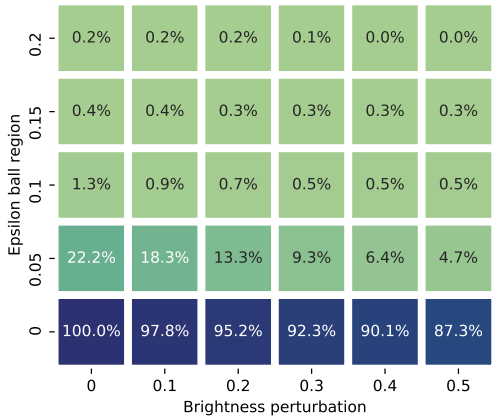


Fig. 3: Percentage of UNSAT queries per noise and brightness parameters.

The incremental verification algorithm invoked the verifier on 13231 queries, whereas the results of 59% of the queries were deduced, using the incremental approach, without additional invocations. Fig. 4 shows the percentage of UNSAT queries for contrast perturbations within the range $[1-\gamma, 1+\gamma]$. The binary search algorithm invoked the verifier 3915 times, whereas the remaining 62% of the queries were deduced without additional invocations. For contrast perturbations, all queries terminated without a timeout.

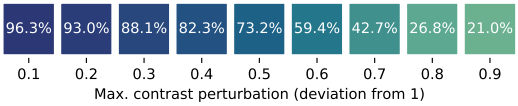


Fig. 4: Percentage of UNSAT queries per contrast parameter

Overall, the results indicate that the classifier shows similar robustness to contrast and brightness perturbations. However, it is significantly more sensitive to noise perturbations. We note that noise in images comes from various sources. Some noise is inherent to the camera's sensor (e.g., impulse noise, thermal noise) or its associated electronics (e.g. shot noise). Other noise is a direct consequence of operating and environmental conditions (e.g., low-light conditions, scenery colors, etc.). Brightness and contrast also fall into this category; they are both inherently related to operating conditions. Although noise originating from image acquisition is certainly a nuisance, it can in part be reduced by noise reduction techniques, as well as an expert understanding of the camera characteristics and continuous quality tracking. Other kinds of noise are more challenging to predict or mitigate, as the number of different operating conditions (weather, time of day, scenery, etc.) is effectively infinite. Therefore, it is somehow reassuring that our classifier seems to be less vulnerable to contrast and brightness, as these perturbations are highly unpredictable.

VI. Conclusion

As numerous state-of-the-art image classifiers are vulnerable to small image perturbations, robustness is a key safety re-

quirement; and certification authorities, such as EASA, might require confirmed robustness as part of the model certification process in the aerospace domain.[7] This work explores the challenges that the industry is facing in its effort to safely deploy deep-learning-based systems, and the benefits that formal methods can afford in assessing the robustness of DNN models to various perturbations. One significant challenge is the limited scalability of current verification techniques.

In this work we focused on assessing the robustness of a prototype runway object classifier provided by Airbus, with respect to three common image perturbations types. To partially address the scalability challenge, we exploited the monotonicity of these perturbations in designing an algorithm that improved the performance of the overall verification process. Moving forward, we aim to assess additional, larger, Airbus networks with higher-resolution input; and to verify their robustness to simultaneous brightness and contrast perturbations. To improve performance, which will enable verifying larger networks, we intend to examine applying DNN abstraction methods [3] to the verification queries we have used. In addition, we aspire to increase the reliability of the results by using the *proof producing* version of Marabou [10].

Acknowledgements. This research was partially funded by Airbus Central Research & Technology, AI Research.

References

[1] Brix, C., Müller, M., Bak, S., Johnson, T., Liu, C.: First Three Years of the International Verification of Neural Networks Competition (VNN-COMP). Int. Journal on Software Tools for Technology Transfer pp. 1–11 (2023)

[2] Casadio, M., Komendantskaya, E., Daggitt, M., Kokke, W., Katz, G., Amir, G., Refaeli, I.: Neural Network Robustness as a Verification Property: A Principled Case Study. In: Proc. 34th Int. Conf. on Computer Aided Verification (CAV). pp. 219–231 (2022)

[3] Elboher, Y., Gottschlich, J., Katz, G.: An Abstraction-Based Framework for Neural Network Verification. In: Proc. 32nd Int. Conf. on Computer Aided Verification (CAV). pp. 43–65 (2020)

[4] Elboher, Y., Elsaleh, R., Isac, O., Ducoffe, M., Galametz, A., Povéda, G., Boumazouza, R., Cohen, N., Katz, G.: Robustness Assessment of a Runway Object Classifier for Safe Aircraft Taxiing (2024), technical Report. http://arxiv.org/abs/2402.00035

[5] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)

[6] Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples (2014), technical Report. http://arxiv.org/abs/1412.6572

[7] Gopinath, D., Katz, G., Păsăreanu, C., Barrett, C.: DeepSafe: A Data-driven Approach for Assessing Robustness of Neural Networks. In: Proc. 16th. Int. Symposium on Automated Technology for Verification and Analysis (ATVA). pp. 3–19 (2018)

[8] Henriksen, P., Lomuscio, A.: Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search. In: Proc. 24th European Conf. on Artificial Intelligence (ECAI). pp. 2513–2520 (2020)

[9] Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety Verification of Deep Neural Networks. In: Proc. 29th Int. Conf. on Computer Aided Verification (CAV). pp. 3–29 (2017)

[10] Isac, O., Barrett, C., Zhang, M., Katz, G.: Neural Network Verification with Proof Production. In: Proc. 22nd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD). pp. 38–48 (2022)

[11] Katz, G., Barrett, C., Dill, D., Julian, K., Kochenderfer, M.: Reluplex: a Calculus for Reasoning about Deep Neural Networks. Formal Methods in System Design (FMSD) (2021)

[12] Lyu, Z., Ko, C.Y., Kong, Z., Wong, N., Lin, D., Daniel, L.: Fastened Crown: Tightened Neural Network Robustness Certificates. In: Proc. 34th AAAI Conf. on Artificial Intelligence (AAAI). pp. 5037–5044 (2020)

[13] Mangal, R., Nori, A., Orso, A.: Robustness of Neural Networks: A Probabilistic and Practical Approach. In: Proc. IEEE/ACM 41st Int. Conf. on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). pp. 93–96 (2019)

[14] Müller, M., Makarchuk, G., Singh, G., Püschel, M., Vechev, M.: PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations. In: Proc. 49th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL) (2022)

[15] Ostrovsky, M., Barrett, C., Katz, G.: An Abstraction-Refinement Approach to Verifying Convolutional Neural Networks. In: Proc. 20th. Int. Symposium on Automated Technology for Verification and Analysis (ATVA). pp. 391–396 (2022)

[16] Singh, G., Gehr, T., Puschel, M., Vechev, M.: An Abstract Domain for Certifying Neural Networks. In: Proc. 46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL) (2019)

[17] Szegedy, C., Toshev, A., Erhan, D.: Deep Neural Networks for Object Detection. Advances in Neural Information Processing Systems 26 (2013)

[18] Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M., Kokke, W., Refaeli, I., Amir, G., Julian, K., Bassan, S., Huang, P., Lahav, O., Wu, M., Zhang, M., Komendantskaya, E., Katz, G., Barrett, C.: Marabou 2.0: A Versatile Formal Analyzer of Neural Networks. In: Proc. 36th Int. Conf. on Computer Aided Verification (CAV) (2024)

[19] Wu, H., Zeljić, A., Katz, G., Barrett, C.: Efficient Neural Network Analysis with Sum-of-Infeasibilities. In: Proc. 28th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 143–163 (2022)

[7] https://www.easa.europa.eu/en/easa-concept-paper-first-usable-guidance-level-1- machine-learning-applications-proposed-issue-01pdf

Appendix

A. Visualization of Brightness and Contrast Perturbations

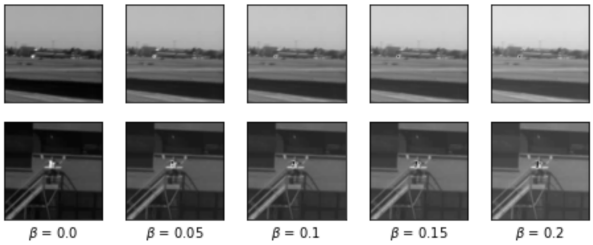


Fig. 5: Brightness perturbations for an 'Aircraft' and a 'Person' from the test set.

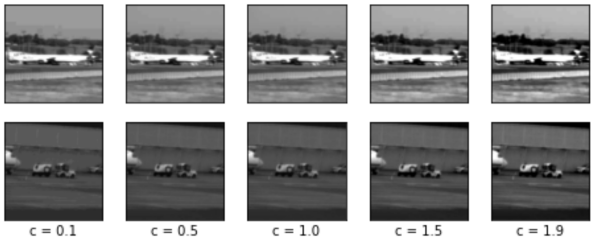


Fig. 6: Contrast perturbations for an 'Aircraft' and a 'Vehicle' from the test set.

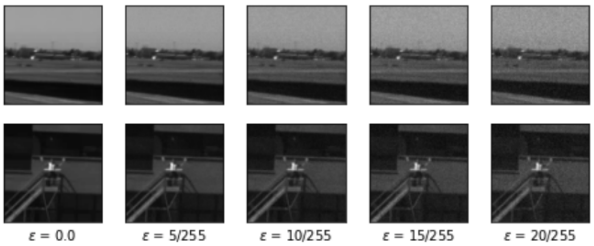


Fig. 7: Levels of l_∞-norm bounded perturbations for an 'Aircraft' and a 'Vehicle'.

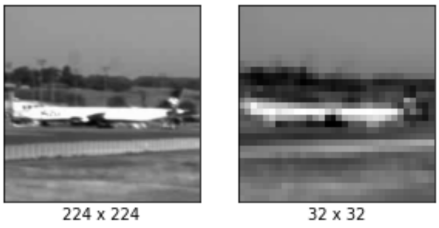


Fig. 8: Illustration of an 'Aircraft' image at different resolutions.

B. An Example of DNN

Consider the DNN with 4 layers that appears in Fig. 9, where all biases are set to zero and are ignored. For input $\langle 2, -1\rangle$, the first node in the second layer evaluates to $\text{ReLU}(2 \cdot 1.5 + -1 \cdot (-1)) = \text{ReLU}(4) = 4$; and the second node in the second layer evaluates to $\text{ReLU}(2 \cdot -1) = \text{ReLU}(-2) = 0$; Then the node in the third layer evaluates to $\text{ReLU}(4-0) = 4$. and thus the output of the network is 2.

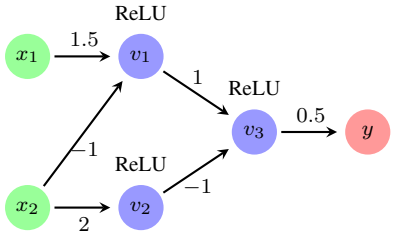


Fig. 9: A toy DNN.