A Proofs

Lemma 1. *Let ϕ be a conjunction of linear inequalities over the variables x_i for i from 0 to $n-1$. We can construct an H-polytope $H = (A, b)$ with Alg. 2 s.t. $(Ax \leq b) \Leftrightarrow (x \models \phi)$.*

Proof. Let $f(x) = \sum_0^{n-1} a_i x_i$ We first show that every lin. ineq. in the conjunction can be reformulated to the form $f(x) \leq b$. It is trivial to show the ineq. can be manipulated to have variables on lhs and a constant on rhs, that $\geq$ can be manipulated to an equivalent form with $\leq$, and $>$ can be manipulated to become $<$. The $<$ comparison can be changed to a $\leq$ comparison by decrementing the rhs constant from b to b' where b' is the largest representable number less than b. We prove ineq. with $<$ can be reformulated to use $\leq$ by contradiction. Assume either $f(x) < b$ and $f(x) > b'$ or $f(x) \geq b$ and $f(x) \leq b'$. Either $b' < f(x) < b$, a contradiction, since $f(x)$ cannot be both larger than the largest representable number less than b and also less than b.[1] Or $b \leq f(x) \leq b'$, a contradiction, since $b' < b$ by definition.

Given a conjunction of lin. ineq. in the form $f(x) \leq b$, Alg. 2 constructs A and b with a row in A and value in b corresponding to each conjunct. There are two cases: $(Ax \leq b) \rightarrow (x \models \phi)$ and $(x \models \phi) \rightarrow (Ax \leq b)$.

We prove case 1 by contradiction. Assume $(Ax \leq b)$ and $(x \not\models \phi)$. By construction of H in Alg. 2, each conjunct of ϕ is exactly 1 constraint in H. If $Ax \leq b$, then all constraints in H must be satisifed, and thus all conjuncts in ϕ must be satisfied and $x \models \phi$, a contradiction.

We prove case 2 by contradiction. Assume $(x \models \phi)$ and $(Ax \not\leq b)$. By construction of H in Alg. 2, each conjunct of ϕ is exactly 1 constraint in H. If $x \models \phi$, then all conjuncts in ϕ must be satisfied, and thus all constraints in H must be satisifed and $Ax \leq b$, a contradiction. □

Lemma 2. *Let $H = (A, b)$ be an H-polytope s.t. $Ax \leq b$. Alg. 4 constructs a DNN, $\mathcal{N}_s$, that classifies whether inputs satisfy $Ax \leq b$. Formally, $x \in H \Leftrightarrow \mathcal{N}_s(x)_0 \leq \mathcal{N}_s(x)_1$.*

Proof. There are 2 cases:

1. $x \in H \rightarrow \mathcal{N}_s(x)_0 \leq \mathcal{N}_s(x)_1$
2. $\mathcal{N}_s(x)_0 \leq \mathcal{N}_s(x)_1 \rightarrow x \in H$

We prove case 1 by contradiction. Assume $x \in H$ and $\mathcal{N}_s(x)_0 > \mathcal{N}_s(x)_1$. From Alg. 4, each neuron in the hidden layer of $\mathcal{N}_s$ corresponds to one constraint in H. The weights of each neuron are the values in the corresponding row of A, and the bias is the negation of the corresponding value of b. If input x satisfies the constraint, then the neuron value will be at most 0, otherwise it will be greater than 0. After the ReLU, each neuron will be equal to 0 if the corresponding constraint is satisfied by x and greater than 0 otherwise. The first output neuron sums all neurons in the hidden layer, while the second has a constant value of 0. If $x \in H$, then all neurons in the hidden layer after activation must have a value of 0 since all constraints are satisfied. However, if all neurons have value 0, then their sum must also be 0, and therefore $\mathcal{N}_s(x)_0 = \mathcal{N}_s(x)_1$, a contradiction.

We prove case 2 by contradiction. Assume $\mathcal{N}_s(x)_0 \leq \mathcal{N}_s(x)_1$ and $x \notin H$. If $x \notin H$, at least one neuron in the hidden layer must have a value greater than 0 after the ReLU since at least one constraint is not satisfied. Because some neuron has a value greater than 0, their sum must also be greater than 0, and therefore $\mathcal{N}_s(x)_0 > \mathcal{N}_s(x)_1$, a contradiction. □

Lemma 3. *Let $H = (A, b)$ be an H-polytope s.t. $Ax \leq b$. Alg. 3 constructs a DNN, $\mathcal{N}_p$, that maps values from the n-dim. unit hypercube to the axis aligned hyperrectangle that minimally bounds H. The range of this mapping does not exclude any x s.t. $Ax \leq b$. Formally, $\forall x \in H. \exists z \in [0,1]^n. x = \mathcal{N}_p(z)$.*

Proof. The proof is by contradiction. Let the axis aligned hyperrectangle that minimally bounds H be specified by lower bounds $\vec{lb}$ and upper bounds $\vec{ub}$ s.t. $\forall x \in H. \forall i. x_i \in [lb_i, ub_i]$. Alg. 3 constructs a DNN, $\mathcal{N}_p$, that computes $\mathcal{N}_p(z) = Wz + b$, where $W = \text{diag}(ub - lb)$ and $b = lb$. This function is invertible: $\mathcal{N}_p^{-1}(x) = W^{-1}(x - b) = W^{-1}x - W^{-1}b$. Assume $\exists x \in H. \exists i. (z = \mathcal{N}_p^{-1}(x)) \wedge ((z_i < 0) \vee (z_i > 1))$. From the def. of $\mathcal{N}_p^{-1}$, we get $\mathcal{N}_p^{-1}(lb)_i \leq z_i \leq \mathcal{N}_p^{-1}(ub)_i$ and $W_{i,i}^{-1}(lb_i) - W_{i,i}^{-1}(lb_i) = 0 \leq z_i \leq W_{i,i}^{-1}(ub_i) - W_{i,i}^{-1}(lb_i) = (\frac{1}{ub_i - lb_i}(ub_i) - \frac{1}{ub_i - lb_i}(lb_i)) = 1$. Therefore $(lb_i \leq x_i \leq ub_i) \rightarrow (0 \leq z_i \leq 1)$, a contradiction. □

[1] We discuss the assumption that such a number exists in Appendix A.1.

Theorem 1. *Let $\psi = \langle \mathcal{N}, \phi \rangle$ be a correctness problem with its property defined as a formula of disjunctions and conjunctions of linear inequalities over the inputs and outputs of $\mathcal{N}$. Property Reduction (Alg. 1) maps ψ to an equivalid set of correctness problems $reduce(\psi) = \{\langle \mathcal{N}_1, \phi_1 \rangle, \ldots, \langle \mathcal{N}_k, \phi_k \rangle\}$.*

$$\mathcal{N} \models \psi \Leftrightarrow \forall \langle \mathcal{N}_i, \phi_i \rangle \in reduce(\psi). \mathcal{N}_i \models \phi_i$$

Proof. A model that satisfies any disjunct of $DNF(\neg\phi)$ falsifies ϕ. If ϕ is falsifiable, then at least one disjunct of $DNF(\neg\phi)$ is satisfiable.

Alg. 1 constructs a correctness problem for each disjunct. For each disjunct, Alg. 1 constructs an H-polytope, H, which is used to construct a prefix network, $\mathcal{N}_p$, and suffix network, $\mathcal{N}_s$. The algorithm then constructs networks $\mathcal{N}'(x) = concat(\mathcal{N}(x), x)$ and $\mathcal{N}''(x) = \mathcal{N}_s(\mathcal{N}'(\mathcal{N}_p(x)))$. Alg. 1 pairs each constructed network with the property $\phi = \forall x. x \in [0,1]^n \rightarrow \mathcal{N}''(x)_0 > \mathcal{N}''(x)_1$. A violation occurs only when $\mathcal{N}''(x)_0 \leq \mathcal{N}''(x)_1$. By Lemmas 1, 2, and 3, we get that $\mathcal{N}''(x)_0 \leq \mathcal{N}''(x)_1$ if and only if $\mathcal{N}'(x) \in H$. If $\mathcal{N}'(x) \in H$ then $\mathcal{N}'(x)$ satisfies the disjunct and is therefore a violation of the original property. □

A.1 On Existance of a Bounded Largest Representable Number

Our proof that property reduction generates a set of robustness problems equivalid to an arbitrary problem relies on the assumption that strict inequalities can be converted to non-strict inequalities. To do so we rely on the existance of a largest representable number that is less than some given value. While this is not necessarily true for all sets of numbers (e.g., $\mathbb{R}$), it is true for for most numeric representations used in computation (e.g., IEEE 754 floating point arithmetic).

B Benchmarks

B.1 ACAS Xu Property Benchmark

The ACAS Xu problem benchmark consists of 10 DNN properties, each applied to a subset of 45 small networks. This benchmark is described in detail in Appendix VI of [1]. Each of the 45 fully-connected networks in this benchmark have 5 input values and 5 output values with 6 hidden layers of 50 neurons each and relu activations. For completeness, we provide formal definitions of the 10 ACAS Xu properties.

Property ϕ_1

$$\begin{aligned}\forall x.((55947.691 \leq x_0 \leq 60760) \wedge (-\pi \leq x_1 \leq \pi) \wedge (-\pi \leq x_2 \leq \pi)\\ \wedge(1145 \leq x_3 \leq 1200) \wedge (0 \leq x_4 \leq 60)) \rightarrow (\mathcal{N}(x)_0 \leq 1500)\end{aligned}$$

Property ϕ_2

$$\begin{aligned}\forall x.((55947.691 \leq x_0 \leq 60760) \wedge (-\pi \leq x_1 \leq \pi) \wedge (-\pi \leq x_2 \leq \pi)\\ \wedge(1145 \leq x_3 \leq 1200) \wedge (0 \leq x_4 \leq 60)) \rightarrow (argmax(\mathcal{N}(x)) \neq 0)\end{aligned}$$

Property ϕ_3

$$\begin{aligned}\forall x.((1500 \leq x_0 \leq 1800) \wedge (-0.06 \leq x_1 \leq 0.06) \wedge (3.10 \leq x_2 \leq \pi)\\ \wedge(980 \leq x_3 \leq 1200) \wedge (960 \leq x_4 \leq 1200)) \rightarrow (argmin(\mathcal{N}(x)) \neq 0)\end{aligned}$$

Property ϕ_4

$$\begin{aligned}\forall x.((1500 \leq x_0 \leq 1800) \wedge (-0.06 \leq x_1 \leq 0.06) \wedge (0 \leq x_2 \leq 0)\\ \wedge(1000 \leq x_3 \leq 1200) \wedge (700 \leq x_4 \leq 800)) \rightarrow (argmin(\mathcal{N}(x)) \neq 0)\end{aligned}$$

Property ϕ_5

$$\forall x.((250 \leq x_0 \leq 400) \wedge (0.2 \leq x_1 \leq 0.4) \wedge (-\pi \leq x_2 \leq -\pi + 0.005) \\ \wedge (100 \leq x_3 \leq 400) \wedge (0 \leq x_4 \leq 400)) \rightarrow (argmin(\mathcal{N}(x)) = 4)$$

Property ϕ_6

$$\forall x.(((12000 \leq x_0 \leq 62000) \wedge (0.7 \leq x_1 \leq \pi) \wedge (-\pi \leq x_2 \leq -\pi + 0.005) \\ \wedge (100 \leq x_3 \leq 1200) \wedge (0 \leq x_4 \leq 1200)) \\ \vee (12000 \leq x_0 \leq 62000) \wedge (-\pi \leq x_1 \leq -0.7) \wedge (-\pi \leq x_2 \leq -\pi + 0.005) \\ \wedge (100 \leq x_3 \leq 1200) \wedge (0 \leq x_4 \leq 1200)) \rightarrow (argmin(\mathcal{N}(x)) = 0)$$

Property ϕ_7

$$\forall x.((0 \leq x_0 \leq 60760) \wedge (-\pi \leq x_1 \leq \pi) \wedge (-\pi \leq x_2 \leq \pi) \\ \wedge (100 \leq x_3 \leq 1200) \wedge (0 \leq x_4 \leq 1200)) \rightarrow (argmin(\mathcal{N}(x)) \neq 4)$$

Property ϕ_8

$$\forall x.((0 \leq x_0 \leq 60760) \wedge (-\pi \leq x_1 \leq -0.75\pi) \wedge (-0.1 \leq x_2 \leq 0.1) \\ \wedge (600 \leq x_3 \leq 1200) \wedge (600 \leq x_4 \leq 1200)) \rightarrow ((argmin(\mathcal{N}(x)) = 0) \vee (argmin(\mathcal{N}(x)) = 1))$$

Property ϕ_9

$$\forall x.((2000 \leq x_0 \leq 7000) \wedge (-0.4 \leq x_1 \leq -0.14) \wedge (-\pi \leq x_2 \leq -\pi + 0.01) \\ \wedge (100 \leq x_3 \leq 150) \wedge (0 \leq x_4 \leq 150)) \rightarrow (argmin(\mathcal{N}(x)) = 3)$$

Property ϕ_{10}

$$\forall x.((36000 \leq x_0 \leq 60760) \wedge (0.7 \leq x_1 \leq \pi) \wedge (-\pi \leq x_2 \leq -\pi + 0.01) \\ \wedge (900 \leq x_3 \leq 1200) \wedge (600 \leq x_4 \leq 1200)) \rightarrow (argmin(\mathcal{N}(x)) = 0)$$

B.2 Neurify-DAVE Property Benchmark

The Neurify-DAVE benchmark, introduced in [4], is a set of local interval-reachability properties applied to a network that predicts steering angles for a self-driving car. The original benchmark applied these properties to a smaller version of the original DAVE DNN. The networks take 100x100 color images as input and produce a single value, y, which is converted to a value between $-\pi$ and π with the function $f(x) = 2 * arctan(x)$. While the smaller network has an input domain of $[0, 1]^{30000}$, the original network uses an input domain of $[-103.939, 103.939]^{10000} \times [-116.779, 116.779]^{10000} \times [-123.68, 123.68]^{10000}$, due to mean centering of inputs originally in the interval $[0, 255]^{30000}$.

The small version of DAVE has 2 convolutional layers with relu activations, and 24 and 36 5x5 kernels, respectively. Both of these layers use strides of 5 and have no padding. These are followed by 2 fully-connected layers, the first of which has a size of 100 and relu activations, and the second of which has a single neuron and no activation. This network has 10277 neurons and 81261 parameters. In addition to this

small network, we include the original DAVE network as part of this benchmark to help demonstrate the scalability of analyses. The original DAVE networks has five convolutional layers with 24, 26, 48, 64, and 64 convolutional kernels, respectively. The first 3 layers use 5x5 kernels, with strides of 2, while the next two use 3x3 kernels with strides of 1. All of the convolutional layers use relu activations and have no padding. The convolutional layers are followed by 5 fully-connected layers with sizes 1164, 100, 50, 10, 1, respectively. The first four of these have relu activations. The original DAVE network has 82669 neurons and 2116983 parameters.

The properties for the Neurify-DAVE benchmark all have the following form: for all inputs within distance ε from input x, the output value must be within 15 degrees of $\mathcal{N}(x)$. Formally, this can be stated as:

$$\forall x'.((x' \in [x-\varepsilon, x+\varepsilon]) \wedge (x' \in \mathcal{X})) \rightarrow (\mathcal{N}(x) - 15^\circ \leq \mathcal{N}(x') \leq \mathcal{N}(x) + 15^\circ)$$

where $\mathcal{X}$ is the appropriate input domain, described above. This benchmark uses $\varepsilon \in \{1, 2, 5, 8, 10\}$ for the original DAVE network, and $\varepsilon \in \{\frac{1}{255}, \frac{2}{255}, \frac{5}{255}, \frac{8}{255}, \frac{10}{255}\}$ for the small network to adjust for input domain.

B.3 GHPR Problem Benchmark

The global halfspace-polytope reachability (GHPR) problem benchmark, is made up of 2 sets of properties, one of which is defined over MNIST networks, and one of which is defined over the DroNet network. Each property sets consists of 10 properties. Within the benchmark, the 10 MNIST properties are each applied to 2 networks, drawn from benchmark used for the ERAN verifier [3]. We chose to use a small convolutional network and a medium convolutional network, both with relu activations. The 10 DroNet properties are applied to the DroNet network [2], which has a ResNet based architecture. The properties are described in more detail below.

B.3.1 MNIST

The networks used as part of the GHPR-MNIST benchmark are the `convSmallRELU__Point.pyt` and `convMedGRELU__Point.pyt` models from the ERAN-MNIST benchmark[2]. The small network has 2 convolutional layers with 16 and 32 4x4 kernels respectively, and strides of 2 and no padding. The convolutional layers are followed by 2 fully-connected layers with dimensions 100 and 10, respectively. The network has 4398 neurons and 89608 parameters. The medium network has 2 convolutional layers with 16 and 32 4x4 kernels respectively, and strides of 2 and uses zero padding. The convolutional layers are followed by 2 fully-connected layers with dimensions 1000 and 10, respectively. The network has 6498 neurons and 1587508 parameters.

The MNIST properties are of the form: for all inputs, the output values for classes a and b are closer to one another than either is to the output value of class c. The values of a, b, and c are selected from the confusion matrix of the medium convolutional network on the MNIST test set, shown in Table 1 with the diagonal values removed. We select the 10 pairs of a and b with the most confusion. We then select a value for c, such that images of digit a were never classified as c, and images of digit b were never classified as c. As an example, we would select 4 and 9 for a and b, since images of fours were classified as nines 13 times, more than any other pair. We then select the value 8 for c, since no images of fours or nines were ever misclassified as eights. This results in 10 properties, defined formally below.

Property ϕ_0.

$$\begin{aligned}\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_4 - \mathcal{N}(x)_9| < |\mathcal{N}(x)_4 - \mathcal{N}(x)_8|) \\ \wedge (|\mathcal{N}(x)_4 - \mathcal{N}(x)_9| < |\mathcal{N}(x)_9 - \mathcal{N}(x)_8|))\end{aligned}$$

Property ϕ_1.

$$\begin{aligned}\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_3 - \mathcal{N}(x)_8| < |\mathcal{N}(x)_3 - \mathcal{N}(x)_1|) \\ \wedge (|\mathcal{N}(x)_3 - \mathcal{N}(x)_8| < |\mathcal{N}(x)_8 - \mathcal{N}(x)_1|))\end{aligned}$$

[2]Available at `https://github.com/eth-sri/eran#neural-networks-and-datasets`

	Predicted Label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	*	1	1	0	1	0	0	0	2	1
1	0	*	1	3	0	1	0	0	0	0
2	1	2	*	1	0	0	1	2	0	0
3	0	0	0	*	0	1	0	2	1	4
4	0	0	1	0	*	0	4	2	0	13
5	2	0	1	10	0	*	1	1	1	1
6	7	3	0	1	2	3	*	0	0	0
7	1	4	7	1	0	0	0	*	1	3
8	4	0	5	10	0	4	0	2	*	5
9	2	3	0	2	4	2	0	3	0	*

Table 1: The confusion matrix of the medium convolutional DNN on the MNIST test set.

Property ϕ_2.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_5 - \mathcal{N}(x)_3| < |\mathcal{N}(x)_5 - \mathcal{N}(x)_4|) \\ \wedge(|\mathcal{N}(x)_5 - \mathcal{N}(x)_3| < |\mathcal{N}(x)_3 - \mathcal{N}(x)_4|))$$

Property ϕ_3.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_6 - \mathcal{N}(x)_0| < |\mathcal{N}(x)_6 - \mathcal{N}(x)_7|) \\ \wedge(|\mathcal{N}(x)_6 - \mathcal{N}(x)_0| < |\mathcal{N}(x)_0 - \mathcal{N}(x)_7|))$$

Property ϕ_4.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_7 - \mathcal{N}(x)_2| < |\mathcal{N}(x)_7 - \mathcal{N}(x)_5|) \\ \wedge(|\mathcal{N}(x)_7 - \mathcal{N}(x)_2| < |\mathcal{N}(x)_2 - \mathcal{N}(x)_5|))$$

Property ϕ_5.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_8 - \mathcal{N}(x)_9| < |\mathcal{N}(x)_8 - \mathcal{N}(x)_6|) \\ \wedge(|\mathcal{N}(x)_8 - \mathcal{N}(x)_9| < |\mathcal{N}(x)_9 - \mathcal{N}(x)_6|))$$

Property ϕ_6.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_8 - \mathcal{N}(x)_2| < |\mathcal{N}(x)_8 - \mathcal{N}(x)_4|) \\ \wedge(|\mathcal{N}(x)_8 - \mathcal{N}(x)_2| < |\mathcal{N}(x)_2 - \mathcal{N}(x)_4|))$$

Property ϕ_7.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_7 - \mathcal{N}(x)_1| < |\mathcal{N}(x)_7 - \mathcal{N}(x)_6|) \\ \wedge(|\mathcal{N}(x)_7 - \mathcal{N}(x)_1| < |\mathcal{N}(x)_1 - \mathcal{N}(x)_6|))$$

Property ϕ_8.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_3 - \mathcal{N}(x)_9| < |\mathcal{N}(x)_3 - \mathcal{N}(x)_2|) \\ \wedge(|\mathcal{N}(x)_3 - \mathcal{N}(x)_9| < |\mathcal{N}(x)_9 - \mathcal{N}(x)_2|))$$

Property ϕ_9.

$$\forall x.(x \in [0,1]^n) \rightarrow ((|\mathcal{N}(x)_8 - \mathcal{N}(x)_5| < |\mathcal{N}(x)_8 - \mathcal{N}(x)_4|) \\ \wedge(|\mathcal{N}(x)_8 - \mathcal{N}(x)_5| < |\mathcal{N}(x)_5 - \mathcal{N}(x)_4|))$$

B.3.2 DroNet

The network used for the GHPR-DroNet benchmark is the DroNet network[3] [2] for autonomous quadrotor control. This network is based on a ResNet type architecture, with 3 residual blocks. It is comprised of 475131 neurons and 320226 parameters.

The properties for DroNet codify the desired behavior that, if the probability for collision is low, the system should not make sharp turns. The DroNet properties are of the form: for all inputs, if the probability of collision is between p_{min} and p_{max}, then the steering angle is within d degrees of 0.

Property ϕ_0.

$$\forall x.((x \in [0,1]^n) \wedge (0 < \mathcal{N}(x)_P \leq 0.1)) \rightarrow (-5^\circ \leq \mathcal{N}(x)_S \leq 5^\circ)$$

Property ϕ_1.

$$\forall x.((x \in [0,1]^n) \wedge (0.1 < \mathcal{N}(x)_P \leq 0.2)) \rightarrow (-10^\circ \leq \mathcal{N}(x)_S \leq 10^\circ)$$

Property ϕ_2.

$$\forall x.((x \in [0,1]^n) \wedge (0.2 < \mathcal{N}(x)_P \leq 0.3)) \rightarrow (-20^\circ \leq \mathcal{N}(x)_S \leq 20^\circ)$$

Property ϕ_3.

$$\forall x.((x \in [0,1]^n) \wedge (0.3 < \mathcal{N}(x)_P \leq 0.4)) \rightarrow (-30^\circ \leq \mathcal{N}(x)_S \leq 30^\circ)$$

Property ϕ_4.

$$\forall x.((x \in [0,1]^n) \wedge (0.4 < \mathcal{N}(x)_P \leq 0.5)) \rightarrow (-40^\circ \leq \mathcal{N}(x)_S \leq 40^\circ)$$

Property ϕ_5.

$$\forall x.((x \in [0,1]^n) \wedge (0.5 < \mathcal{N}(x)_P \leq 0.6)) \rightarrow (-50^\circ \leq \mathcal{N}(x)_S \leq 50^\circ)$$

Property ϕ_6.

$$\forall x.((x \in [0,1]^n) \wedge (0.6 < \mathcal{N}(x)_P \leq 0.7)) \rightarrow (-60^\circ \leq \mathcal{N}(x)_S \leq 60^\circ)$$

Property ϕ_7.

$$\forall x.((x \in [0,1]^n) \wedge (0.7 < \mathcal{N}(x)_P \leq 0.8)) \rightarrow (-70^\circ \leq \mathcal{N}(x)_S \leq 70^\circ)$$

Property ϕ_8.

$$\forall x.((x \in [0,1]^n) \wedge (0.8 < \mathcal{N}(x)_P \leq 0.9)) \rightarrow (-80^\circ \leq \mathcal{N}(x)_S \leq 80^\circ)$$

Property ϕ_9.

$$\forall x.((x \in [0,1]^n) \wedge (0.9 < \mathcal{N}(x)_P \leq 1.0)) \rightarrow (-90^\circ \leq \mathcal{N}(x)_S \leq 90^\circ)$$

[3] `https://github.com/uzh-rpg/rpg_public_dronet`

B.4 CIFAR-EQ Property Benchmark

The CIFAR-EQ problem benchmark, is made up of a set of 291 equivalence properties defined over 2 networks trained on CIFAR10. The benchmark has a 91 global equivalence properties, the first of which is an untargeted equivalence property specifying that the two networks must predict the same class for every input.

$$\forall x.(x \in [0,1]^n) \rightarrow (argmax(\mathcal{N}_1(x)) = argmax(\mathcal{N}_2(x)))$$

The other 90 properties are targeted equivalence properties, specifying that if the first network predicts class A, then the second network cannot predict class B, and vice versa, where A and B are different classes. We create a property for each possible pair of output classes for a total of 90 properties.

$$\forall x.(x \in [0,1]^n) \rightarrow ((argmax(\mathcal{N}_1(x)) \neq A) \vee (argmax(\mathcal{N}_2(x)) \neq B))$$

The next 200 properties are local properties, created from the first 10 images from the CIFAR10 test set that are correctly classified by both networks. Each local property is specified with an L_∞ ϵ-ball around the original input. In this work, we use the epsilon values of $\frac{1}{255}$ and $\frac{10}{255}$. The first 20 properties are untargeted equivalence properties, specifying that all inputs within the ϵ-ball are classified as the same class by both networks. This results in 20 properties, 2 for each of the 10 inputs.

$$\forall x'.(x' \in [x-\epsilon, x+\epsilon]^n) \rightarrow (argmax(\mathcal{N}_1(x')) = argmax(\mathcal{N}_2(x')))$$

The final 180 properties are targeted equivalence properties, specifying that if either network classifies the input to the correct class, C, then the other network should not classify it as class D, different from the correct class. This results in 180 properties, 18 for each of the 10 inputs.

$$\begin{aligned}\forall x'.(x' \in [x-\epsilon, x+\epsilon]^n) \rightarrow (((argmax(\mathcal{N}_1(x')) = C) \rightarrow (argmax(\mathcal{N}_2(x')) \neq D)) \\ \wedge\ ((argmax(\mathcal{N}_2(x')) = C) \rightarrow (argmax(\mathcal{N}_1(x')) \neq D)))\end{aligned}$$

References

[1] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 97–117, 2017.

[2] Antonio Loquercio, Ana Isabel Maqueda, Carlos R. Del Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 2018.

[3] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10802–10813. Curran Associates, Inc., 2018.

[4] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *NeurIPS*, pages 6369–6379, 2018.