Appendix: DNNV

David Shriver, Matthew B. Dwyer, and Sebastian Elbaum

University of Virginia, Charlottesville, VA, USA
{dls2fc,matthewbdwyer,selbaum}@virginia.edu

A Verification Benchmarks

We examine the benchmarks used to evaluate each of the 13 verifiers supported by DNNV, and determine whether each verifier can run on the benchmark out of the box, and also whether they could be run on the benchmark when DNNV is applied. Here we provide a short description of each of the 19 verification benchmarks that we have identified. A short summary of some of the features of each verifier relevant to DNNV are shown in Table 1. These features include whether any properties cannot represent their input constraints using hyper-rectangles ($\neg$HR), whether any network in the benchmark contains convolution operations (C), whether any network contains residual structures (R), and whether any network uses any non-ReLU activation functions ($\neg$ReLU).

The ACAS Xu (AX) benchmark, introduced for *Reluplex* [7], is one of the most used verification benchmarks [4, 16, 8, 2]. The benchmark consists of 10 properties. Property ϕ_1 is a reachability property, specifying an upper bound on one of the 5 output variables. Properties ϕ_5, ϕ_6, ϕ_9, and ϕ_{10} are all traditional class robustness properties, specifying the desired class for the given input region. Properties ϕ_3, ϕ_4, ϕ_7 and ϕ_8 are reachability properties, specifying a set of acceptable classes for the input region. Properties ϕ_2 is also a reachability property, specifying that a given output value cannot be greater than all others. Each of the properties are applied to a subset of 45 networks trained on an aircraft collision avoidance dataset, with 5 inputs, 5 output classes and 6 layers of 50 neurons each. The original benchmark included networks in *Reluplex*-NNET format, and a custom version of *Reluplex* was written for each property. Later uses of the benchmark translated the verification problems into RLV format, which is used by *Planet*, *BaB*, and *BaBSB*, as well as translating the networks into ONNX. The benchmark in ONNX and DNNP format is fully supported by DNNV.

The Collision Detection (CD) benchmark [5], intoduced for the evaluation of *Planet*, consists of 500 local robustness properties for an 80 neuron network with a fully connected layer and max pooling layer that classifies whether 2 simulated vehicles will collide, given their current state. The verification problems, in RLV format, are supported by *Planet*, *BaB*, and *BaBSB*. The problems have also been modified to convert max pooling operations to a sequence of fully-connected layers with ReLU activations, and then translated to *Reluplex*-NNET format, enabling off the shelf support by *Marabou*, and a generalized version of *Reluplex*. This benchmark is one of the few that is not supported by DNNV,

Table 1. Verifier benchmarks.

					Features			
Key	**Name**	**Uses**	**#P**	**#$\mathcal{N}$**	**¬HR**	**C**	**R**	**¬ReLU**
AX	ACAS Xu	[7, 4, 16, 8, 2]	10	45				
CD	Collision Detection	[5, 4, 8]	500	1				
PM	*Planet* MNIST	[5]	7	1	✓	✓		
TS	TwinStream	[3]	1	81				
PCA	PCAMNIST	[4]	12	17				
MM	*MIPVerify* MNIST	[15]	10000	5		✓		
MC	*MIPVerify* CIFAR10	[15]	10000	2		✓	✓	
NM	*Neurify* MNIST	[16, 6]	500	4		✓		
NDB	*Neurify* Drebin	[16]	500	3				
NDV	*Neurify* DAVE	[16]	200	1	✓	✓		
DZM	*DeepZono* MNIST	[12]	1700	10		✓	✓	✓
DZC	*DeepZono* CIFAR10	[12]	1700	5		✓		✓
DPM	*DeepPoly* MNIST	[13, 6]	1500	8		✓		✓
DPC	*DeepPoly* CIFAR10	[13]	800	5		✓		
RZM	*RefineZono* MNIST	[14]	800	8		✓		
RZC	*RefineZono* CIFAR10	[14]	200	2		✓		
RPM	*RefinePoly* MNIST	[11]	600	6		✓		
RPC	*RefinePoly* CIFAR10	[11]	300	3		✓	✓	
VC	*VeriNet* CIFAR10	[6]	250	1		✓		

since the network contains structures that are not easily supported by ONNX. In particular, the max-pooling operation in the original network, applied to a flat tensor, cannot be encoded by ONNX from their original format.

The *Planet* MNIST (PM) benchmark [5] is a set of 7 properties over a convolutional network trained on the MNIST dataset [10]. The first 4 of these are reachability properties with hyper-rectangle input constraints, while the next 2 are local robustness properties with hyper-rectangle input constraints, and the final property is an local robustness property with halfspace-polytope input constraints. The original benchmark was provided in RLV format. The first 6 of these properties are currently supported by DNNV, while the final property could be supported by DNNV with additional engineering effort.

The TwinStream (TS) benchmark [3] consists of 1 property applied to 81 networks that output a constant value. The property asserts that for all inputs, the output of the network is positive. The original benchmark was provided in RLV format. This benchmark is fully supported by DNNV for all verifiers.

The PCAMNIST (PCA) benchmark [4] consists of 12 reachability properties applied to 17 networks trained on modified versions of the MNIST dataset to predict the parity of the digit represented by the first k components of the PCA decomposition of an image. The original benchmark was provided in RLV format. This benchmark is fully supported by DNNV for all verifiers.

MIPVerify MNIST (MM) consists of 10000 local robustness properties applied to 5 networks trained on the MNIST dataset. The networks have varied structures: 2 networks are fully connected and 3 are convolutional. We could not find an available version of the benchmark used by *MIPVerify* to evaluate its original input format. This benchmark is fully supported by DNNV for all verifiers except *Reluplex*, which does not support convolution operations.

MIPVerify CIFAR (MC) consists of 10000 local robustness properties applied to 2 networks trained on the CIFAR10 dataset [9]. One of these networks is a convolutional network and the other is a residual network. We could not find an available version of the benchmark used by *MIPVerify* to evaluate its original input format. This benchmark is supported by DNNV for verifiers that can support residual connections, including: *Planet*, *DeepZono*, *DeepPoly*, *RefineZono*, and *RefinePoly*. While the benchmark is supported by the version of *MIPVerify* used in its study, it is not supported through DNNV, since the publicly available version of *MIPVerify* does not support residual connections.

The *Neurify* MNIST (NM) benchmark [16] consists of 500 L_∞ local robustness properties across 4 MNIST networks, 3 fully connected networks with 58, 110, and 1034 neurons respectively, and a convolutional network with 4814 neurons. The original benchmark was provided in *Neurify*-NNET format, with properties hard-coded into the verifier. DNNV enables almost all verifiers to run on this benchmark. *Reluplex* cannot be run due to the presence of convolutional layers, which are not supported. *MIPVerify* cannot be run due to the presence of non-hypercube input constraints. While this limitation of the verifier can be satisfied with a property reduction for fully-connected networks, DNNV does not currenly support such a reduction for convolutional networks.

The *Neurify* Drebin (NDB) benchmark [16] consists of 500 L_∞ local robustness properties across 3 fully connected Drebin [1] networks with 102, 212, and 402 neurons each. The original benchmark was provided in *Neurify*-NNET format, with properties hard-coded into the verifier. This benchmark is fully supported by DNNV for all verifiers.

The *Neurify* DAVE (NDV) benchmark [16] consists of 200 local reachability properties, with 4 different types of input constraints (50 properties of each type). The first type of input constraint is an L_∞ constraint, which is equivalent to a hyper-rectangle constraint. The second type of input constraint is an L_1 constraint, which can be written as a halfspace polytope constraint. The third and fourth type of input constraint are image brightning and contrast, which can be written as halfspace polytope constraints. The properties are applied to a convolutional network for an autonomous vehicle, with 10276 neurons. The original benchmark was provided in *Neurify*-NNET format, with properties hard-coded into the verifier. Similar to the *Neurify* MNIST benchmark, DNNV enables almost all verifiers to run on this benchmark. *Reluplex* cannot be run, due to the presence of convolutional layers, which are not supported, and *MIPVerify* cannot be run due to the presence of non-hypercube input constraints.

The *DeepZono* MNIST (DZM) benchmark [12] consists of 1700 local robustness properties, subsets of which are applied to 10 networks trained on the MNIST dataset. The networks have varied structures and activation functions: 3 networks are fully connected, 1 of which uses ReLU activations, 1 with Tanh activations, and 1 with Sigmoid activations; 6 are convolutional, 4 of which have ReLU activations, 1 with Tanh activations, and 1 with Sigmoid activations; and 1 is a residual network. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV does not increase the support for this benchmark due to the presence of both a residual network and non-ReLU activation functions.

The *DeepZono* CIFAR10 (DZC) benchmark [12] consists of 1700 local robustness properties, subsets of which are applied to 5 networks trained on the CIFAR10 dataset. The networks have varied structures and activation functions: 3 networks are fully connected, 1 of which uses ReLU activations, 1 with Tanh activations, and 1 with Sigmoid activations; and 2 are convolutional with ReLU activations. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables *VeriNet* to run on this benchmark. Other verifiers are not supported due to the non-ReLU activation functions.

The *DeepPoly* MNIST (DPM) benchmark [13] consists of 1500 local robustness properties, subsets of which are applied to 8 networks trained on the MNIST dataset. The networks have varied structures and activation functions: 5 networks are fully connected, 3 of which uses ReLU activations, 1 with Tanh activations, and 1 with Sigmoid activations; and 3 are convolutional with ReLU activations. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables *VeriNet* to run on this benchmark. Other verifiers are not supported due to the non-ReLU activation functions.

The *DeepPoly* CIFAR10 (DPC) benchmark [13] consists of 800 local robustness properties, subsets of which are applied to 5 networks trained on the CIFAR10 dataset. The networks have varied structures: 3 networks are fully connected with ReLU activations; and 2 are convolutional with ReLU activations. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables several additional verifiers to support this benchmark. In particular, it enables most verifiers that can be applied to convolutional networks with relu activations.

The *RefineZono* MNIST (RZM) benchmark [14] consists of 800 local robustness properties, subsets of which are applied to 8 networks trained on the MNIST dataset. 5 networks are fully connected with ReLU activations and 3 are convolutional with ReLU activations. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables several additional verifiers to support this benchmark. In particular, it enables most verifiers that can be applied to convolutional networks with relu activations.

The *RefineZono* CIFAR10 (RZC) benchmark [14] consists of 200 local robustness properties, subsets of which are applied to 2 networks trained on the CIFAR10 dataset. One of the networks is fully connected with ReLU activations and the other is convolutional with ReLU activations. The networks in the originalinal benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables several additional verifiers to support this benchmark. In particular, it enables most verifiers that can be applied to convolutional networks with relu activations.

The *RefinePoly* MNIST (RPM) benchmark [11] consists of 600 local robustness properties, subsets of which are applied to 6 networks trained on the MNIST dataset. 4 networks are fully connected with ReLU activations and 2 are convolutional with ReLU activations. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables several additional verifiers to support this benchmark. In particular, it enables most verifiers that can be applied to convolutional networks with relu activations.

The *RefinePoly* CIFAR10 (RPC) benchmark [11] consists of 300 local robustness properties, subsets of which are applied to 3 networks trained on the MNIST dataset. Two of the networks are convolutional with ReLU activations and the third is a residual network with ReLU activations. The networks in the original benchmark were provided in a custom human-readable text format, with properties hard-coded into the verifier. DNNV enables the *Planet* verifier to support this benchmark. In particular, it enables most verifiers that can be applied to convolutional networks with relu activations. Other verifiers do not support the residual structure of one of the networks.

The *VeriNet* CIFAR10 (VC) benchmark [6] consists of 250 local robustness properties applied to 1 convolutional network with ReLU activations. The networks were provided in ONNX format, with hard-coded properties. DNNV enables support of this benchmark by most of the integrated verifiers. *Reluplex* does not support convolutional networks, and *MIPVerify* does not support properties with input constraints that are not hyper-cubes.

A.1 Support

We summarize the support of each verifier for each of the benchmarks in Table 2. Each row of this table corresponds to one of the 13 verifiers supported by DNNV, and each column corresponds to one of the 19 benchmarks identified in Table 1. Each cell of the table may contain a circle that identifies the support of the verifier for the benchmark. The left half of the circle is filled black if the verifier can support the benchmark out of the box, and is filled white otherwise. The right half is filled black if the verifier supports the benchmark through DNNV, filled gray if support is planned, and is filled white otherwise. Planned support means that DNNV will support the benchmark after the implementation for halfspace polytope constraints in the input space is completed. We plan to implement support for this feature by the notification deadline on December 23, 2020. An absent circle indicates that the verifier can not be made to support some aspect of the benchmark. For the benchmarks shown here, this is always due to the presence of non-ReLU activation functions in some of the networks in the benchmarks.

Table 2. Benchmark support by each verifier. The left half of the circle is black if the verifier can support the benchmark out of the box, and is white otherwise. The right half is black if the verifier supports the benchmark through DNNV, gray if support is pending DNNV implementation, and is white otherwise. An absent circle indicates that the verifier can not be made to support some aspect of the benchmark.

	Benchmark																		
Verifier	AX	CD	PM	TS	PCA	MM	MC	NM	NDB	NDV	DZM	DZC	DPM	DPC	RZM	RZC	RPM	RPC	VC
Reluplex	●	◐	○	●	●	○	○	○	◑	○				○	○	○	○	○	○
Planet	●	◐	◐ (gray)	●	●	◑	◑	◑	◑	◑ (gray)				◑	◑	◑	◑	◑	◑
BaB	●	◐	◐ (gray)	●	●	◑	○	◑	◑	◑ (gray)				◑	◑	◑	◑	○	◑
BaBSB	●	◐	◐ (gray)	●	●	◑	○	◑	◑	◑ (gray)				◑	◑	◑	◑	○	◑
MIPVerify	◑	○	○	◑	◑	●	◐	○	◑	○				○	○	○	○	○	○
Neurify	●	○	◑ (gray)	◑	◑	◑	○	●	●	◐ (gray)				◑	◑	◑	◑	○	◑
DeepZono	●	○	◑ (gray)	◑	◑	◑	◑	◑	◑	◑ (gray)	●	●	●	●	●	●	●	●	◑
DeepPoly	●	○	◑ (gray)	◑	◑	◑	◑	◑	◑	◑ (gray)	●	●	●	●	●	●	●	●	◑
RefineZono	●	○	◑ (gray)	◑	◑	◑	◑	◑	◑	◑ (gray)	●	●	●	●	●	●	●	●	◑
RefinePoly	●	○	◑ (gray)	◑	◑	◑	◑	◑	◑	◑ (gray)	●	●	●	●	●	●	●	●	◑
Marabou	●	◐	◑ (gray)	●	●	◑	○	◑	◑	◑ (gray)				◑	◑	◑	◑	○	◑
nnenum	●	○	◑ (gray)	◑	◑	◑	○	◑	◑	◑ (gray)				◑	◑	◑	◑	○	◑
VeriNet	◑	○	◑ (gray)	◑	◑	◑	○	●	◑	◑ (gray)	○	◑	◑	◑	◑	◑	◑	○	●

References

1. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K.: DREBIN: effective and explainable detection of android malware in your pocket. In: 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014. The Internet Society (2014), `https://www.ndss-symposium.org/ndss2014/drebin-effective-and-explainable-detection-android-malware-your-pocket`
2. Bak, S., Tran, H.D., Hobbs, K., Johnson, T.T.: Improved geometric path enumeration for verifying relu neural networks. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification. pp. 66–96. Springer International Publishing, Cham (2020)
3. Bunel, R., Turkaslan, I., Torr, P.H.S., Kohli, P., Kumar, M.P.: Piecewise linear neural network verification: A comparative study. CoRR **abs/1711.00455v1** (2017), `http://arxiv.org/abs/1711.00455v1`
4. Bunel, R.R., Turkaslan, I., Torr, P.H.S., Kohli, P., Mudigonda, P.K.: A unified view of piecewise linear neural network verification. In: NeurIPS. pp. 4795–4804 (2018)
5. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings. pp. 269–286 (2017). https://doi.org/10.1007/978-3-319-68167-2_19, `https://doi.org/10.1007/978-3-319-68167-2\_19`
6. Henriksen, P., Lomuscio, A.: Efficient neural network verification via adaptive refinement and adversarial search. In: Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) ECAI 2020 - 24th European

Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 2513–2520. IOS Press (2020). https://doi.org/10.3233/FAIA200385, https://doi.org/10.3233/FAIA200385

7. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: 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. pp. 97–117 (2017). https://doi.org/10.1007/978-3-319-63387-9_5, https://doi.org/10.1007/978-3-319-63387-9_5
8. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The Marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification. pp. 443–452. Springer (2019)
9. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
10. LeCun, Y., Cortes, C., Burges, C.J.: The mnist database of handwritten digits
11. Singh, G., Ganvir, R., Püschel, M., Vechev, M.T.: Beyond the single neuron convex barrier for neural network certification. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. pp. 15072–15083 (2019), http://papers.nips.cc/paper/9646-beyond-the-single-neuron-convex-barrier-for-neural-network-certification
12. Singh, G., Gehr, T., Mirman, M., Püschel, M., Vechev, M.: Fast and effective robustness certification. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 10802–10813. Curran Associates, Inc. (2018), http://papers.nips.cc/paper/8278-fast-and-effective-robustness-certification.pdf
13. Singh, G., Gehr, T., Püschel, M., Vechev, M.T.: An abstract domain for certifying neural networks. PACMPL **3**(POPL), 41:1–41:30 (2019)
14. Singh, G., Gehr, T., Püschel, M., Vechev, M.T.: Boosting robustness certification of neural networks. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=HJgeEh09KQ
15. Tjeng, V., Xiao, K.Y., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=HyGIdiRqtm
16. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. In: NeurIPS. pp. 6369–6379 (2018)