Deeper Notions of Correctness in Image-Based DNNs: Lifting Properties from Pixel to Entities*

Felipe Toledo
University of Virginia
Charlottesville, VA, USA
ft8bn@virginia.edu

David Shriver
University of Virginia
Charlottesville, VA, USA
dls2fc@virginia.edu

Sebastian Elbaum
University of Virginia
Charlottesville, VA, USA
selbaum@virginia.edu

Matthew B. Dwyer
University of Virginia
Charlottesville, VA, USA
dwyer@virginia.edu

ABSTRACT

Deep Neural Networks (DNNs) that process images are being widely used for many safety-critical tasks, from autonomous vehicles to medical diagnosis. Currently, DNN correctness properties are defined at the pixel level over the entire input. Such properties are useful to expose system failures related to sensor noise or adversarial attacks, but they cannot capture features that are relevant to domain-specific entities and reflect richer types of behaviors. To overcome this limitation, we envision the specification of properties based on the entities that may be present in image input, capturing their semantics and how they change. Creating such properties today is difficult as it requires determining where the entities appear in images, defining how each entity can change, and writing a specification that is compatible with each particular V&V client. We introduce an initial framework structured around those challenges to assist in the generation of Domain-specific Entity-based properties automatically by leveraging object detection models to identify entities in images and creating properties based on entity features. Our feasibility study provides initial evidence that the new properties can uncover interesting system failures, such as changes in skin color can modify the output of a gender classification network. We conclude by analyzing the framework potential to implement the vision and by outlining directions for future work.

CCS CONCEPTS

• **Software and its engineering** → **Software verification and validation**; **Correctness**.

KEYWORDS

Neural networks, validation, verification, properties, fairness

ACM Reference Format:
Felipe Toledo, David Shriver, Sebastian Elbaum, and Matthew B. Dwyer. 2023. Deeper Notions of Correctness in Image-Based DNNs: Lifting Properties from Pixel to Entities. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3611643.3613079

*This work was supported by NSF grants #2019239 and #2332991.

ESEC/FSE '23, December 3–9, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0327-0/23/12.
https://doi.org/10.1145/3611643.3613079

1 INTRODUCTION

The increasing capability of DNN to synthesize accurate implementations for challenging problems has led to their deployment in high-consequence domains, such as airport security screening [24], medical diagnosis [36], US criminal justice system [38], or autonomous systems [3, 4, 19]. For such domains where failures can lead to significant consequences, it is necessary to check the extent to which a system satisfies its intended properties.

To address this need, researchers have developed a body of DNN validation and verification (V&V) techniques. Since the seminal Reluplex paper [15] that adapted SMT-based verification to reason about DNNs to check for property satisfaction, dozens of DNN verifiers have been developed, e.g., [8, 34, 37, 41]. Within the space of validation techniques, falsifiers such as adversarial attacks [10, 18, 20, 31, 32] have been developed to quickly find counter example to DNN robustness properties. In addition, many testing approaches have been introduced [42], among those, structurally-driven techniques [23, 26, 33] have been developed to generate inputs aimed at satisfying a DNN coverage criteria, while model-driven have emerged to generate inputs better suited for a particular domain [7, 27, 45].

In spite of the noticeable progress in V&V techniques for DNNs, the properties targeted have remained mostly limited to pixel constraints and applied to the whole input space. Consider a self-driving vehicle governed by a DNN that consumes sensed images and produces a steering angle for the vehicle. One could check a reachability property like *"independent of the input image the steering angle should not exceed a given threshold"*. Now consider a DNN consuming photos and detecting if there are people or not. A robustness property for such DNN may state that *"small variations in image pixel values should not alter the classification"*. These properties are appealing because they can be easily defined over the whole input and applied to *all* images in a dataset, and they can assess whether the DNN is resilient to sensor noise, variation in lighting, or adversarial attacks.

Many DNN failures, however, are caused by the violation of properties relevant to specific regions of an input associated with domain-specific entities. For instance, in the context of self-driving vehicles, recent fatal crashes have involved image entities such as light-colored trucks [1] or shaded road-barriers [21]. In such cases, being able to specify properties like *"independent of the vehicle color, the steering angle should remain the same"* would have increased the likelihood that such failing behavior is exposed before deployment. Some works have explored entity-based properties, focusing on signs [17, 43]. However, these approaches rely on manual entity detection, which can be time-consuming and error-prone, only detect a limited set of entity types, have been applied

to just one domain, or do not broadly generalize to different change types. Despite these challenges, the specification of entity-based properties can be valuable across many domains. For example, in the context of fairness, several systems have been found to be biased toward certain skin colors, such as the COMPAS system, used by the U.S. courts to predict the tendency of a convicted criminal to re-offend, which has been shown to predict recidivism more frequently for black offenders than white offenders [13]. Similarly, Google Vision Cloud labeled a picture of a dark-skinned arm holding a thermometer as a "gun", whereas a light-skinned arm holding the same thermometer was labeled as an "electronic device" [22], and Google Photos mislabeled a picture of two African Americans as "gorilas" [6]. Being able to specify a property like *"independent of changes to the skin color in an image, the output class of the network should remain the same"* (explored in Section 4), would have enabled existing V&V tools to detect such failures.

While the cases mentioned above involve entity color changes, they only represent a fraction of the possible transformations that entities can undergo in real-world scenarios. For example, let us imagine that we are validating a DNN that classifies people and cars. Changes in style or shape would allow us to define interesting properties like *"independent of the car's model, the predicted class should not change"* or *"regardless of straight or curly hairstyle, the predicted class should not change"*. Further, changing the composition of an image, e.g., the location or pose of an entity, would allow us to define richer properties like *"independent of the location and pose of a person on the sidewalk, the predicted class should not change"*.

VISION. Our vision is to enable the specification of Domain-specific Entity-based (DSEB) properties that capture the semantics of the entities in the world and how they can change. These properties would enable different V&V clients to explore the wide space of entity variations and check if DNNs are invariant to changes that may occur in the real world, thus assessing their robustness and generalization capabilities. Consequently, stronger guarantees will ensure that models in production perform well on unseen data. Ultimately, the goal is to deploy more reliable and trustworthy AI systems that can be used in safety-critical applications, making them more useful and beneficial to society.

2 RESEARCH CHALLENGES

To specify DSEB properties, like the ones stated above, to assess DNN's behavior, uncover serious DNN misbehaviors, and deploy more reliable AI systems, engineers face significant challenges.

(1) **Entity identification.** First, they must identify the region corresponding to a target entity, such as a car in a road scene or hair in the image of a person. In principle, this could be performed manually but segmenting a target entity in complex inputs like images is a difficult and costly endeavor. For instance, a human-annotated segmentation can require 5,400 seconds [28], and this identification must be performed for every image since the entity's location and appearance can vary across images.
(2) **Entity transformation.** Second, engineers must specify how that entity can change. There are countless ways that entities might change, following previous examples, style changes could be performed by modifying the make and model of a car, or the hairstyle of a person. Additionally, composition changes could be achieved, by changing the location and pose of cars and people. Nonetheless, these entity transformations are not trivial to automate and would require manual effort, e.g., using photo editing software.
(3) **Encoding.** Lastly, writing specifications is a difficult task, as it requires defining the behavior of a system in a formal language that can be processed by different V&V clients, that have diverse input types and capabilities. For instance, verifiers tend to focus on proving that a property specification holds for a DNN, while validators, such as testing methods or falsifiers, can only prove a property is false by finding a counter example. On the other hand, while verifiers have limited support for network operations, falsifiers are less restrictive but require the network to be differentiable, whereas testing can support any input type. Thus, creating specifications that work with a range V&V clients is challenging.

3 FRAMEWORK FOR REALIZING VISION

In this section, we formally define a correctness problem and sketch a framework for supporting the specification of Entity-based properties with Domain-specific semantics that are amenable to automated verification and validation.

3.1 Correctness problem

A correctness problem is a pair, $\psi = \langle \mathcal{N}, \phi \rangle$, of a DNN, $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, and a property specification ϕ. The property specification defines constraints over the inputs and outputs of $\mathcal{N}$. We further differentiate between constraints over only inputs, $\phi_{\mathcal{X}}$, which we refer to as the pre-condition, and constraints over both inputs and outputs or only outputs, $\phi_{\mathcal{Y}}$, which we refer to as the post-condition. Verification and validation techniques seek to check whether $\forall \boldsymbol{x} \in \mathbb{R}^n : \phi_{\mathcal{X}}(x) \rightarrow \phi_{\mathcal{Y}}(\mathcal{N}(x))$ is true or false.

3.2 Overview

An outline of the envisioned framework, Domain-specific Entity-Based Property Generation (DSEBPropGen), is depicted in Fig. 1. For a network under analysis, $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, DSEBPropGen takes an image, x_0, and outputs a correctness problem. DSEBPropGen has three main components, designed to solve each of the challenges.

First, to address the challenge of identifying entities, *Entity Identification* automatically detects a set of entity types, Σ, in the input, x_0, using a user-configured entity detector. For each instance, i, of an entity type, $\sigma \in \Sigma$, it returns a set of pixels, $R_i^\sigma \subseteq [1, n]$, associated with that entity. For example, the locations of the pixels in a car, face, or hair within an image. Second, to specify how an entity can vary in terms of color, style, location, etc., *Entity Transformation* applies a set of user-configurable functions, $T^\sigma : \mathcal{X} \times P(\mathbb{N}) \rightarrow P(\mathcal{X})$, which map an input, $x_0 \in \mathcal{X}$, and a detected entity, R_i^σ, to a set that defines the space of transformed inputs from x_0. For instance, it permits the pixels associated with a face to vary within a color range corresponding to human skin tones. Lastly, to overcome the difficulty of writing specifications, *Encoding* combines the generated constraints with a user-specified post-condition and expresses them in a form compatible with existing V&V clients. We describe each of these steps in more detail in the next few sections.

3.3 Entity Identification

The goal of this step is to detect within a given input the parts of that input that correspond to a semantically relevant entity. To

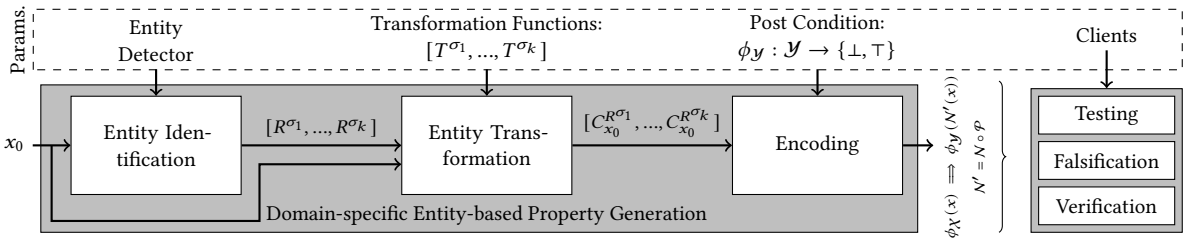


Figure 1: Approach overview.

accomplish this, we propose to leverage automated *entity detectors*. In principle, any automated algorithm capable of detecting a target entity type, σ, with sufficient accuracy can be employed as long as its output can be transformed to R^σ, the set of pixels predicted to contain the target entity. In practice, state-of-the-art pre-trained models for semantic segmentation, e.g., Detectron2 [40], are a scalable and broadly applicable solution. These models also provide mechanisms to control the quality of segmentation, by computing an estimate of their uncertainty, and segmenting instances only if the predicted label's confidence exceeds a certain threshold. For instance, Detectron2 can be used to automatically identify different objects such as cars, trucks, and bicycles, above a confidence threshold, e.g. 0.75 or 0.9 out of 1.

More formally, *Entity Identification* takes in an input x_0 and produces a set R^Σ, such that for each entity type $\sigma \in \Sigma$, $R^\sigma \subseteq R^\Sigma$ identifies the pixels in x_0 that correspond to entity type σ. Multiple entities of a single type may be detected in an input, which we disambiguate with the notation $R^\sigma_i \subseteq R^\sigma$, where $i \in [1, n]$ indicates an id for the entity. We expect that the set of entity types to be identified, Σ could be configured by the user to adjust to their needs.

3.4 Entity Transformation

The goal of this step is to take x_0 and the entities identified by *Entity Identification* and transform x_0 to a set of inputs that allow domain-specific changes to the identified entities. *Entity Transformation* takes an input x_0 and a set of identified entities, $\{R^\sigma_i \mid \sigma \in \Sigma \wedge i \in [1, n]\}$, and generates, for each entity type, σ, a transformation defining the set of inputs that are transformed from x_0 by a user-configured entity-specific transformation function, $T^\sigma : \mathcal{X} \times P(\mathbb{N}) \rightarrow P(\mathcal{X})$. The rest of the input that has not been transformed, $R^{\bar{T}}$, remains invariant. In this work, we use a subscript to differentiate between different transformations, e.g., T^σ_{color} is a function that allows an entity to vary in color in an image, like car color changes in [39]. This transformation can be implemented by adjusting the pixel values to tint them with a desired color. However, more complex transformations such as changing the model of a car would require sophisticated ways of changing the pixel values as we may not only need to change the pixels within the entity, but also surrounding pixels, e.g. if we want to enlarge the entity.

3.5 Encoding

The goal of this step is to take the input sets generated by *Entity Transformation* and encode them in a form supported by V&V clients. *Encoding* takes in a transformed set of inputs, $T(x_0) \subseteq \mathcal{X}$, and a user-configured post-condition specified as a predicate over network outputs $\phi_{\mathcal{Y}} : \mathcal{Y} \rightarrow \{\bot, \top\}$ and outputs a property and network that can be checked by existing V&V tools. The post-condition specifies constraints over the network output, such as the predicted class remaining invariant across the generalized inputs, or the steering angle remaining within some specified range.

The predicate for the transformed input set can be represented as a function $\mathcal{Q} : \mathcal{X} \times P(\mathbb{N}) \times \Gamma \rightarrow \mathcal{X}$, where Γ is a parameter space to the function. The function $\mathcal{Q}$ can be encoded as a neural network and can be used as a prefix $\mathcal{P}$ to the original neural network $\mathcal{N}$, to modify inputs within entity regions with the color transformation. For example for T^σ_{color}, we created a prefix that takes in a color in CMYK format and an intensity value, and modifies the image by changing the entity's color. Using such network prefix as encoding enable the expression of non-linear functions and the flexibility to work with many V&V clients. Users could utilize pre-defined transformation encodings or create new ones. Developing new encodings requires an understanding of the downstream target V&V clients since they have different capabilities. For instance, testing techniques can handle very general network structures, therefore any function can be encoded. On the other hand, property-driven falsifiers require networks comprised of differentiable functions, and verifiers often are limited to a small set of activation functions.

4 FEASIBILITY STUDY

The goal of this study is to explore what it takes to instantiate DSEBPropGen to automatically generate DSEB properties, and the insights those properties can provide in the V&V process.

4.1 Study Setup

Target DNN. We explored the application of DSEBPropGen in the domain of gender classification from face images using a pre-trained ResNet34 [12] provided by FairFace [14].

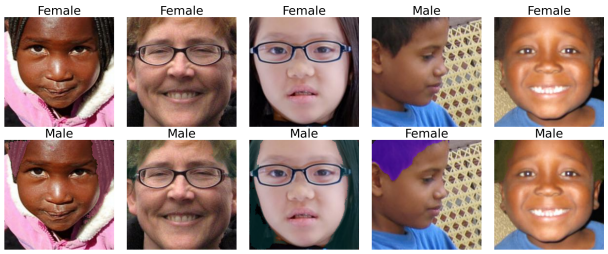
Properties. We were interested in checking 2 properties: (1) *"Gender is independent of skin color"*, and (2) *"Gender is independent of hair color"*. We generated 8,417 properties of each type based on high-quality filtered images from the FairFace dataset.

4.2 Instatiating DSEBPropGen

Identification. We integrated two semantic segmentation models into our prototype. First, Detectron2 [40] trained on LVIS [11], which recognizes over 1200 entity types such as cars, signs, fruits, and appliances. Second, MobileNetV2_unet [2] trained to process headshots, capable of segmenting face, hair, and background. By leveraging the extensive entity set of the first segmentation model, DSEBPropGen can be applied across many domains, while the second model allows for customization to specific domains.

Transformation. We defined a color transformation function, which we applied to two entity types, hair and face, and obtained two different DSEB property types. This transformation includes

(a) Face properties: face color change within a skin-tone.



(b) Hair properties: hair color change.

Figure 2: Counter-examples found by falsifiers at the bottom, with their original image above.

dependencies between pixels so that the color of the entity can change as a whole. Specifically:

$$T^{\sigma}_{\Delta}(x_0, R^{\sigma}) = \{x \in \mathcal{X} \mid x = color(x_0, R^{\sigma}, \delta) \wedge \delta \in \Delta\}$$

where $\delta = (C, M, Y, K, A)$ and *color* is a function that mixes the color of the pixels in R^{σ} in x_0 with the color specified by C, M, Y, and K by an intensity factor A, while leaving the rest of x_0 unchanged. The set Δ can be parameterized to specify different color spaces, which may be useful for different entity types.

We chose different color spaces for the hair and face transformations. For the hair properties, we did not restrict any of the C, M, Y, or K channels since the hair can be of any color. For the face properties, we restricted the channels to represent a set of estimated skin-tone color values [5]. For both property types, we limited the parameter A to not end up having a solid color for the entity.

Encoding. We encoded the color transformation into a DNN prefix, which creates a mask from the input dimensions of the entity and recolors only the pixels of the entity, leaving the rest of the inputs unaltered. We then saved this network in ONNX [25] format and created the property specifications using DNNP [29] format, to make them amenable for use by DNNF [30].

V&V Client. After generating the properties, we used falsification to help us judge the robustness of the target DNN and selected three falsifiers included in DNNF. We chose to use Cleverhans Fast Gradient Method (FGM) [10], Basic Iterative Method (BIM), and Projected Gradient Descent (PGD) [20]. The first 2 falsifiers were run once since they are deterministic, while the non-deterministic PGD method was run repeatedly for 5 minutes.

4.3 V&V of Target Network

For our study, where we have 8,417 seed images for generating DSEB properties, generating human annotations would be cost-prohibitive – ~500 person-days [28]. In contrast, DSEBPropGen can generate DSEB properties for all images in under 30 minutes.

After attempting to falsify all the properties, we found 437 (5.19%) and 502 (5.97%) unique face and hair violations respectively using the 3 falsifiers, indicating that the network is brittle to color changes in some portions of the input. Examples of face and hair violations can be seen in Fig 2a and Fig 2b respectively. In the third column of Fig. 2a we can observe that the face region was segmented correctly, and the color transformation was applied uniformly to all the face pixels, which created a smooth and consistent change in skin tone. Note that the new color falls within the range of natural skin tones, thereby generating a realistic counter-example. Likewise, in the second column of Fig. 2b, we can see that the hair pixels were also detected appropriately, and the resulting color transformation on the hair, which is tinted with green, appears natural. These results highlight the potential to define and check new types of properties that target different entities and go beyond the classical robustness defined over the entire input space.

5 PROGRESS MADE AND FUTURE WORK

We now contrast the vision with the proposed framework to point out gaps that need to be addressed to generate different DSEB properties that enable V&V of a broader range of DNNs behaviors, and enable its application at a system level.

(1) **Detection.** Our vision relies on automatic detection systems to identify entities on which to specify DSEB properties. As discussed in 3.3 and as confirmed by our preliminary study, current segmentation models are fast and capable of recognizing a variety of entities with confidence. These systems, however, can sometimes mispredict an entity or segment it incorrectly. Such errors have different implications depending on the V&V client. For validation techniques, such errors would result in false positives. For a verifier that determines that a property holds, such errors would provide false guarantees. Thus, devising mechanisms that account for such identification errors is going to be required. Furthermore, the sophistication and accuracy of such systems would also have to account for sequences of images [44].

(2) **Transformations.** In our study, we implemented a color transformation that revealed interesting system failures. For example, changes in the skin or hair color that should not modify the network's output were found to do so. However, more complex transformations like changing the model of a car, or the haircut style would require more sophisticated transformations. Recent advances in generative models [9, 16] could help to automate these types of changes, and some efforts [7, 35] have already used them over the entire input space for validating DNN properties. Adapting them to work at the entity level has the potential to enable numerous style transformations. However, even seemingly basic transformations like removing entities remain extremely challenging, as filling in the resulting gaps is a difficult task [39]. Furthermore, transformations that operate consistently across images are not yet supported.

(3) **Encoding.** Using a network prefix to encode transformations can provide flexibility for specifying non-linear changes, and increase the usability of the properties by V&V tools. Nevertheless, creating the prefix currently requires either manual encoding, as we did for the color properties, or either training or finding off-the-shelf models that can handle the desired transformations. Finding ways to further automate this process, or add support to encode broader types of entity transformations remains a significant challenge.

REFERENCES

[1] Yahoo News Aaron Cole. July 1, 2016. Fatal crash in Florida is first reported Tesla Autopilot death. Link.

[2] Akirasosa. 2020. Real-Time Semantic Segmentation in Mobile device. Link.

[3] Long Chen, Shaobo Lin, Xiankai Lu, Dongpu Cao, Hangbin Wu, Chi Guo, Chun Liu, and Fei-Yue Wang. 2021. Deep neural network based vehicle and pedestrian detection for autonomous driving: a survey. *IEEE Transactions on Intelligent Transportation Systems* 22, 6 (2021), 3234–3246.

[4] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. 2018. End-to-End Driving Via Conditional Imitation Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 1–9. https://doi.org/10.1109/ICRA.2018.8460487

[5] Scheme Color. [n. d.]. Skin color palette. Link.

[6] Yahoo News Daniel Howley. June 29, 2015. Google Photos Mislabels 2 Black Americans as Gorillas. Link.

[7] Isaac Dunn, Hadrien Pouget, Daniel Kroening, and Tom Melham. 2021. Exposing previously undetectable faults in deep neural networks. In *ISSTA '21: 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, Denmark, July 11-17, 2021*, Cristian Cadar and Xiangyu Zhang (Eds.). ACM, 56–66. https://doi.org/10.1145/3460319.3464801

[8] Rüdiger Ehlers. 2017. 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*. 269–286. https://doi.org/10.1007/978-3-319-68167-2_19

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[11] Agrim Gupta, Piotr Dollar, and Ross Girshick. 2019. LVIS: A Dataset for Large Vocabulary Instance Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Surya Mattu Julia Angwin, Jeff Larson and ProPublica Lauren Kirchner. April 04, 2023. Machine Bias. Link.

[14] Kimmo Karkkainen and Jungseock Joo. 2021. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1548–1558.

[15] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. 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*. 97–117.

[16] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[17] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. 2020. PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

[19] Antonio Loquercio, Ana Isabel Maqueda, Carlos R. Del Blanco, and Davide Scaramuzza. 2018. Dronet: Learning to Fly by Driving. *IEEE Robotics and Automation Letters* (2018). https://doi.org/10.1109/lra.2018.2795643

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[21] ABC News Mark Osborne. March 31, 2018. Tesla car was on autopilot prior to fatal crash in California, company says. Link.

[22] Algorithm Watch Nicolas Kayser-Bril. April 7, 2020. Google apologizes after its Vision AI produced racist results. Link.

[23] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. 2019. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, California, USA, 4901–4911.

[24] Department of Homeland Security. December 4, 2017. Passenger Screening Algorithm Challenge. Link.

[25] ONNX. 2017. Open Neural Network Exchange. Link.

[26] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. 1–18. https://doi.org/10.1145/3132747.3132785

[27] Vincenzo Riccio and Paolo Tonella. 2020. Model-Based Exploration of the Frontier of Behaviours for Deep Learning System Testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 876–888. https://doi.org/10.1145/3368089.3409730

[28] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. 2016. Playing for data: Ground truth from computer games. In *European conference on computer vision*. Springer, 102–118.

[29] David Shriver, Sebastian G. Elbaum, and Matthew B. Dwyer. 2021. DNNV: A Framework for Deep Neural Network Verification. In *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12759)*, Alexandra Silva and K. Rustan M. Leino (Eds.). Springer, 137–150. https://doi.org/10.1007/978-3-030-81685-8_6

[30] David Shriver, Sebastian G. Elbaum, and Matthew B. Dwyer. 2021. Reducing DNN Properties to Enable Falsification with Adversarial Attacks. In *Proceedings of the International Conference on Software Engineering*.

[31] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2017. One pixel attack for fooling deep neural networks. *CoRR* abs/1710.08864 (2017).

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[33] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. 303–314.

[34] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations*.

[35] Felipe Toledo, David Shriver, Sebastian Elbaum, and Matthew B. Dwyer. 2021. Distribution Models for Falsification and Verification of DNNs. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 317–329. https://doi.org/10.1109/ASE51524.2021.9678590

[36] Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, Venugopalan S, Widner K, Madams T, Cuadros J, Kim R, Raman R, Nelson PC, Mega JL, and Webster DR. December 16, 2016. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs.

[37] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Efficient Formal Safety Analysis of Neural Networks. In *NeurIPS*. 6369–6379.

[38] Wikipedia. May 23, 2016. COMPAS (software). Link.

[39] Trey Woodlief, Sebastian Elbaum, and Kevin Sullivan. 2022. Semantic image fuzzing of AI perception systems. In *Proceedings of the 44th International Conference on Software Engineering*. 1958–1969.

[40] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. Link.

[41] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. *Advances in Neural Information Processing Systems* 31 (2018), 4939–4948.

[42] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* 48, 1 (2022), 1–36. https://doi.org/10.1109/TSE.2019.2962027

[43] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. 2020. DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 347–358.

[44] Tianfei Zhou, Fatih Porikli, David J. Crandall, Luc Van Gool, and Wenguan Wang. 2023. A Survey on Deep Learning Technique for Video Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 6 (2023), 7099–7122. https://doi.org/10.1109/TPAMI.2022.3225573

[45] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. DeepHyperion: Exploring the Feature Space of Deep Learning-Based Systems through Illumination Search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual, Denmark) *(ISSTA 2021)*. Association for Computing Machinery, New York, NY, USA, 79–90.

Received 2023-05-03; accepted 2023-07-19