DeepCert: Verification of Contextually Relevant Robustness for Neural Network Image Classifiers

Colin Paterson[1] ✉, Haoze Wu[2], John Grese[3], Radu Calinescu[1], Corina S. Păsăreanu[3], and Clark Barrett[2]

[1] University of York, York, United Kingdom
✉colin.paterson@york.ac.uk
[2] Stanford University, Stanford, USA
[3] Carnegie Mellon University, Silicon Valley, USA

Abstract. We introduce DeepCert, a tool-supported method for verifying the robustness of deep neural network (DNN) image classifiers to *contextually relevant perturbations* such as blur, haze, and changes in image contrast. While the robustness of DNN classifiers has been the subject of intense research in recent years, the solutions delivered by this research focus on verifying DNN robustness to small perturbations in the images being classified, with perturbation magnitude measured using established L_p norms. This is useful for identifying potential adversarial attacks on DNN image classifiers, but cannot verify DNN robustness to contextually relevant image perturbations, which are typically not small when expressed with L_p norms. DeepCert addresses this underexplored verification problem by supporting: (1) the encoding of real-world image perturbations; (2) the systematic evaluation of contextually relevant DNN robustness, using both testing and formal verification; (3) the generation of contextually relevant counterexamples; and, through these, (4) the selection of DNN image classifiers suitable for the operational context (i) envisaged when a potentially safety-critical system is designed, or (ii) observed by a deployed system. We demonstrate the effectiveness of DeepCert by showing how it can be used to verify the robustness of DNN image classifiers build for two benchmark datasets ('German Traffic Sign' and 'CIFAR-10') to multiple contextually relevant perturbations.

Keywords: Deep neural network robustness · Deep neural network verification · Contextually relevant image perturbations

1 Introduction

Deep neural network (DNN) image classifiers are increasingly being proposed for use in safety critical applications [6,15,19,24], where their accuracy is quoted as close to, or exceeding, that of human operators [3]. It has been shown, however, that when the inputs to the classifier are subjected to small perturbations, even highly accurate DNNs can produce erroneous results [8,9,30]. This has lead to intense research into verification techniques that check whether a DNN is robust

to perturbations within a small distance from a given input, where this distance is measured using an L_p norm (e.g., the Euclidean norm for $p = 2$) [4,12,13,20]. These techniques are particularly useful for identifying potential adversarial attacks on DNNs [8,14,17,18]. They are also useful when small changes in the DNN inputs correspond to meaningful changes in the real world, e.g., to changes in the speed and course of an aircraft for the ACAS Xu DNN verified in [12].

For DNN image classifiers, small L_p-norm image changes are not always meaningful. Changes that may be more meaningful for such DNNs (e.g., image blurring, hazing, variations in lighting conditions, and other natural phenomena) can also cause misclassifications, but are difficult to map to small pixel variations [10,16], and thus cannot be examined using traditional DNN verification techniques. What is needed for the comparison and selection of DNN image classifiers used in safety-critical systems is a *contextually relevant robustness veri cation* method capable of assessing the robustness of DNNs to these real-world phenomena [1,2,25,31]. Moreover, this verification needs to be performed at DNN level (i.e., across large datasets with image samples from all relevant classes) rather than for a single sample image.

The tool-supported DeepCert[4] method introduced in our paper addresses these needs by enabling:

1. The formal encoding of contextually relevant image perturbations at quantified perturbation levels 2 $[0; 1]$.
2. The verification of contextually relevant DNN robustness, to establish how the accuracy of a DNN degrades as the perturbation level increases. DeepCert can perform this verification using either test-based (fast but approximate) or formal verification (slow but providing formal guarantees).
3. The generation of contextually relevant counterexamples. These counterexamples provide engineers with visually meaningful information about the level of blur, haze, etc. at which DNN classifiers stop working correctly.
4. The selection of DNNs appropriate for the operational context (i) envisaged when a safety-critical system is designed, or (ii) observed by the deployed system during operation.

We organised the rest of the paper as follows. Section 2 describes our DeepCert verification method, explaining its encoding of contextual perturbations, and detailing how it can be instantiated to use test-based and formal verification. Section 3 presents the DeepCert implementation, and Section 4 describes the experiments we performed to evaluate it. Finally, Section 5 discusses related work, and Section 6 provides a summary and outlines future research directions.

2 DeepCert verification method

2.1 Overview

Figure 1 shows our DeepCert method for the systematic verification of contextually relevant DNN robustness. DeepCert accepts as input a set of m 1 DNN models, $\bar{M}$, and a dataset of n 1 labelled image samples, . Each element

[4] Deep neural network Contextual robustness

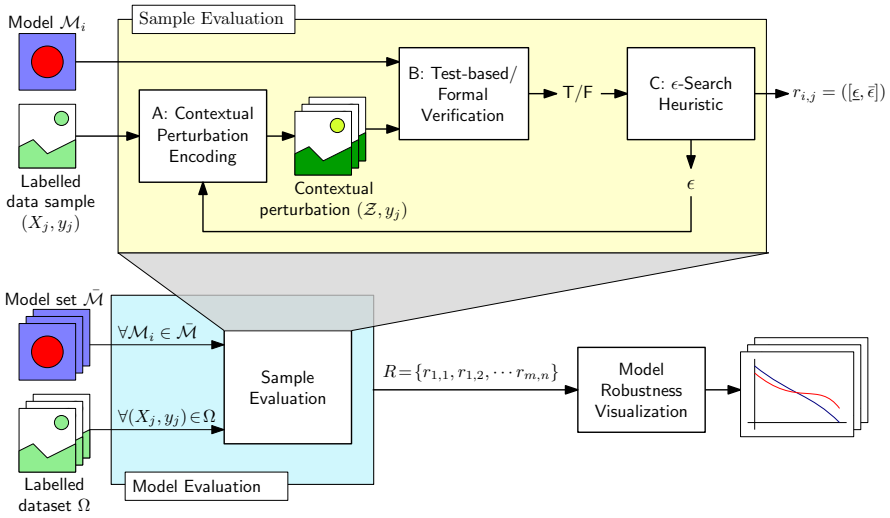


Fig. 1: DeepCert process for verifying contextually meaningful DNN robustness.

$u\ 2$ is a tuple $u = (X;y)$ where $X\ 2\ X$ is the input sample, X is the DNN input space, and y is a label indicating the class into which the models should place the sample. During model evaluation, each model $M_i\ 2\ \bar{M}$ is evaluated against each labelled data sample $(X_j;y_j)\ 2$, to find a robustness measure for that sample. The results are then presented to the engineer as visualisations that enable model-level contextual robustness evaluation and comparison.

The sample evaluation (top of Figure 1) is a three-stage iterative process. The first stage (A) encodes the contextual perturbation using a function $g : X$ $[0;1]\ !\ 2^X$ that maps the data sample $X_j\ 2\ X$ and a *perturbation level* $2\ [0;1]$ to a set of DNN inputs $Z = g(X_j;\ )\ 2\ 2^X$ corresponding to images obtained by applying the contextual perturbation being verified (e.g., haze or blur) to the original image sample X_j. As we explain later in this section, g applies the perturbation at level when DeepCert employs test-based verification, and at *all* levels in the range $[0;\ ]$ when DeepCert employs formal verification.

The second stage (B) verifies whether the model M_i is robust to the contextual perturbation $(Z;y_j)$, i.e., whether it classifies all images from Z as belonging to class y_j. The output of this stage is a Boolean value, true (T) or false (F).

The final state (C) is a search heuristic that supplies the value used for the contextul perturbation encoding from stage A, and employs binary search to identify perturbation level bounds $\_;^- 2\ [0;1]$ such that:

- either $\_ < ^-$, the correct class y_j is predicted for $= \_$, and a misclassification occurs for $= ^-$;
- or $\_ = ^- = 0$, and the DNN misclassifies X_j (with no perturbation applied).

After checking whether X_j is classified correctly by model M_i, the search heuristic starts with $\_ = 0$ and $^- = 1$, halves the width of the interval $[\_;^-]$ in each iteration, and terminates when the width $^- \_$ of this interval drops below a predefined value $!$. The final interval $r_{i;j} = [\_;^-]$ is then returned.

Applying sample evaluation to each model $M_i \in M$ and every sample $X_j \in$ provides a result set $R = \{r_{1;1}; r_{1;2}; \quad r_{m;n}\}$, where $r_{i;j}$ is the interval for the i-th model and j-th image sample. For each result, a counterexample X'_j can be generated, if one exists (i.e., if $\_ < 1$), by perturbing the sample X_j at level $= \bar{}$. Evaluating X'_j using model M_i produces a misclassification label $\hat{y}_j$.

Visualisations of model and class robustness are then produced in which the accuracy of the models is presented as a function of the perturbation parameter . By examining the accuracy of models across the range of expected perturbations, we can identify the conditions under which model switch should occur, e.g. one model may perform well at low levels of haze whilst a second may be superior as the level of haze present increases. Where the visualisations indicate that a particular class accuracy is highly sensitive to changes in this may indicate the need to choose a less sensitive model, or to gather additional training data.

2.2 DeepCert instantiation for test-based verification

For test-based verification, the contextual perturbation encoding function g maps an image X to a set Z comprising a single modified image X' obtained by applying a perturbation function:

$$x'_{i;j} = \mathit{perturbation}(X_{i;j}; \ ); \tag{1}$$

where $x'_{i;j}$ is the pixel at position $(i; j)$ in the modified image X' and $X_{i;j}$ is a subset of pixels from the original image X. For colour images, a sample X is encoded as an array of pixels each of which is a 3-tuple of values representing the red, green and blue components of the colour in that pixel. DeepCert may utilize any perturbation which can be coded using (1) and three typical contextual perturbations are shown in (Figure 2).

Haze encoding. Haze represents a phenomenon where particles in the atmosphere scatter the light reaching the observer. The effect is to drain colour from the image and create a veil of white, or coloured, mist over the image. While realistic approaches to the modelling of haze require complex models [32], simplifying assumptions can be made. Assuming the haze is uniform, a haze colour may be defined as $C^f = (r; g; b)$ and applied to the image as:

$$x'_{i;j} = (1 \quad)x_{i;j} + \quad C^f \tag{2}$$

where $\in [0; 1]$ is a proxy for the density of the haze. When $= 0$ the image is unaltered and when $= 1$ the image is a single solid colour C^f. Multiplication and addition are applied to the pixel in an element-wise manner.

Contrast variation encoding. When fixed aperture lenses are employed, or when the dynamic range of the scene is extreme, the contrast in the image may become compressed. This effect may be modelled as:

$$x'_{i;j} = \mathrm{Max} \quad 0; \mathrm{Min} \quad 1; \frac{x_{i;j} \quad (0{:}5 \quad)}{1} \tag{3}$$

The effect of applying this function is to make bright parts of the image lighter and dark parts of the image darker.

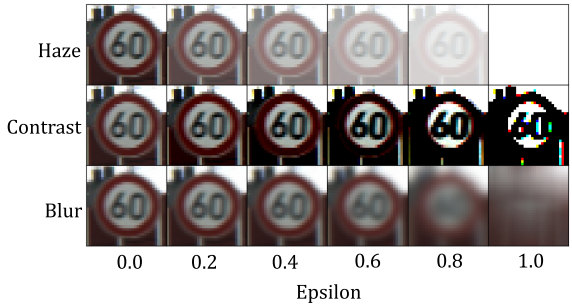


Fig. 2: Context perturbations applied to image sample

Blur encoding. Blurring in an image occurs when parts of the image are out of focus due to the limited capabilities of the optics employed in the system or when grease or water droplets are present on the lens. Blur can be synthesised using a convolutional kernel of size $2k_d + 1$ where the value of a pixel in the output image is calculated as a weighted sum of neighbouring pixels:

$$x'_{i;j} = \sum_{k=-k_d}^{k_d} \sum_{l=-k_d}^{k_d} {}_{k;l} \, x_{i+k;j+l} \tag{4}$$

The weights ${}_{k;l} \in (0;1)$ are calculated by discretising a two-dimensional Gaussian curve, where the sum of weights is equal to one, $\sum_{k=-k_d}^{k_d} \sum_{l=-k_d}^{k_d} {}_{k;l} = 1$. In our work, we define to be proportional to the standard deviation of the Gaussian distribution across the kernel and calculate the weights accordingly.

2.3 DeepCert instantiation for formal verification

While test-based verification is computationally efficient, this efficiency is obtained by sacrificing completeness, i.e. if the perturbed image corresponding to an value of p is not an adversarial example, we cannot guarantee that the network is robust against all perturbations with smaller than p. Formal verification tools, by contrast, can provide such guarantees, but typically impose constraints on the types of models and perturbations which can be analysed.

To demonstrate the use of formal verification within DeepCert, we integrated it with Marabou [13], a complete verification toolbox for analyzing DNNs. Marabou handles common piecewise linear activation functions (e.g., ReLU, Max-Pool, Sign), integrates multiple state-of-the-art bound tightening techniques [21,26,28], and supports parallel processing [29]. Given a neural network and a verification query, Marabou constructs a set of linear and piecewise linear constraints. The satisfiability of the conjunction of those constraints is evaluated using either an MILP-solver or the Reluplex procedure [12]. Given sufficient time, Marabou will either conclude that the query is unsatisfiable or return a satisfying assignment to the query. For this work we extended Marbou to allow for the encoding of contextual perturbations using an input perturbation function, as detailed below for haze.

Haze encoding. Given a DNN model M, an image X, a fog colour C^f, and a maximum perturbation bound p, we introduce variables $X;Y$ and , denoting

the DNN inputs, the DNN outputs and the perturbation bound, respectively. X has the same shape as X. We then construct the following set of constraints:

$$Y = M(X) \tag{5a}$$

$$0 \quad p \tag{5b}$$

$$\bigwedge_{i \; jXj} x_i = (1 \quad) x_i + \quad C^f \tag{5c}$$

$$\bigvee_{\substack{i<=jYj \\ y_i 6= y_{real}}} y_i \quad y_{real} \tag{5d}$$

Checking the satisfiability of the constraints allows us to state if the network is robust against the haze perturbation for p. Constraint (5a) denotes the relationship between X and Y. It is a piecewise linear constraint if M only contains piecewise linear activation functions. Constraint (5b) represents the perturbation bounds. Constraint (5c) defines the input variables as results of the hazing perturbation. Finally, let y_{real} be the correct label, constraint (5d) denotes that the output variable corresponding to the correct label is not greater than that of some other label. The network is locally adversarially robust against haze perturbation with p if, and only if, the conjunction of the constraints above is unsatisfiable. If the constraints above is satisfiable, there exists a perturbation within such that some output other than y_{real} is maximal.

3 Implementation

We implemented our method using a Python framework which we have made available on our tool website https://deepcert.github.io. The repository includes all models used in the paper, the code for the DeepCert tool with the encoded perturbations presented in the paper, the supporting scripts required to generate the performance visualisations and instructions on how to use the framework. In addition, a version of Marabou is provided with a Python interface in which the haze perturbation from the previous section is encoded.

4 Experimental Results

4.1 Case Study 1: Road Tra c Speed Sign Classi cation

Our first case study uses a subset of the German Traffic Sign benchmark [22] where each sample is a 32 32 RGB image. From this set we selected the seven classes which represented speed signs, the number of samples in each class are shown in Table 1a. We then built classification models at three levels of complexity with two models per level. The accuracy for all six models is reported in Table 1b which shows accuracy increasing with model complexity.

DeepCert with test-based veri cation. For each model we applied our method using test-based verification, an initial value of = 0:5 and a binary search heuristic with a maximum permissible interval of 0.002. Figure 3 shows the impact of haze on model accuracy as is increased. While Table 1b shows model 3A to be the most accurate (0.988) without perturbation, we note that

Table 1: German Speed Sign Classification: Data and Models

(a) Data Sets

Class	Description	# Train	# Test
0	30 km/h	1980	720
1	50 km/h	2010	750
2	60 km/h	1260	450
3	70 km/h	1770	660
4	80 km/h	1650	630
5	100 km/h	1290	450
6	120 km/h	1260	450

(b) Models

Model	Description	Accuracy
1A	Small ReLu only model	0.816
1B		0.847
2A	Large ReLu only model	0.868
2B		0.866
3A	CNN Model	0.988
3B		0.984

for $\gtrapprox 0.7$, model 3B achieves superior accuracy. This behaviour is more clearly seen if we consider the ReLu-only models. Here model 2A has the best initial performance, but this rapidly deteriorates as increases such that other models are superior for even small amounts of haze.

These results demonstrate the dangers of selecting a model on the basis of the accuracy reported for unperturbed samples, and show how DeepCert enables a more meaningful model selection for the operational context. Indeed, were the system to be equipped with additional sensing, to assess the level of haze present, the engineer may choose to switch between models as the level of haze increased.

Our method also allows for the identification of those classes particularly susceptible to contextual perturbations. Figure 4 shows the performance of the convolutional neural network (CNN) models at different levels of perturbation. We note that class 1 is largely insensitive to haze, this is because an image perturbed with $= 1$ results in a solid colour image which is classified as class 1 by both models. For all other classes the accuracy reduces as haze increases. The amount of degradation is seen to be dependent on the sample class and the model used. For example, class 0 is more robust to haze in model 3B than in 3A with class 3 more robust in model 3A.

Figure 4c and 4d show the distribution of values required to cause misclassification where circles indicates samples identified as outliers. For class 3 we see that a number of samples are misclassified for small perturbations using model

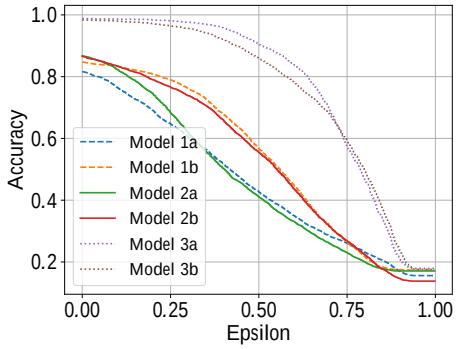
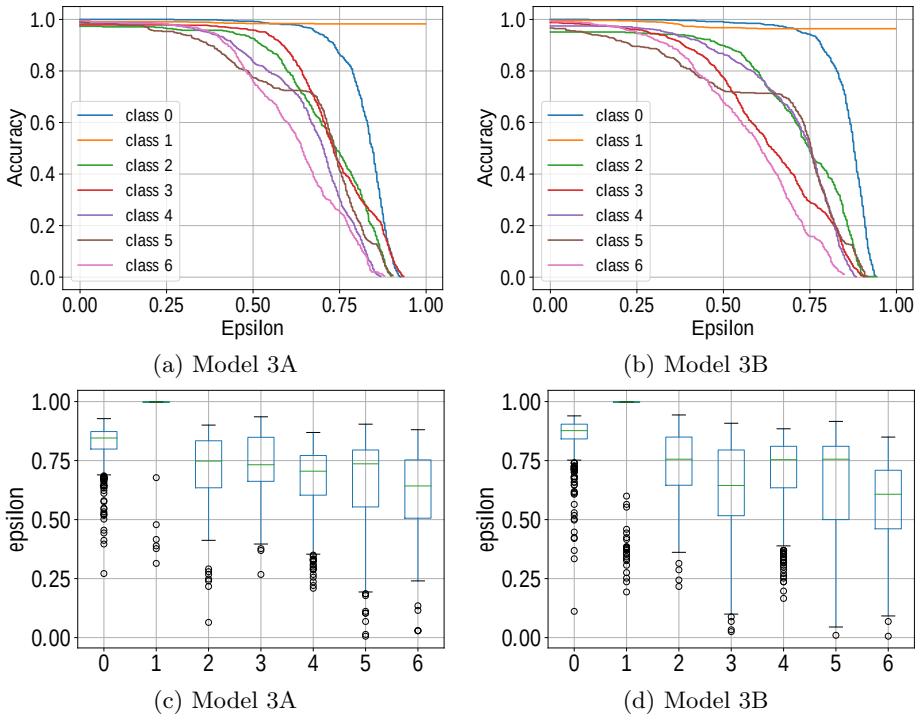


Fig. 3: Model robustness to haze.



(a) Model 3A (b) Model 3B

(c) Model 3A (d) Model 3B

Fig. 4: Model Robustness with respect to haze

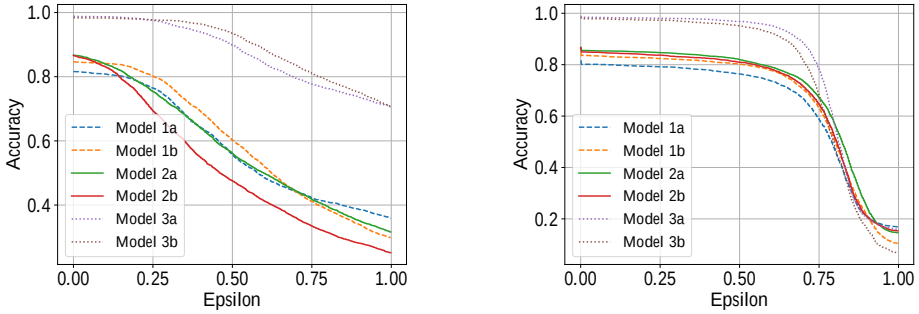
3B but not 3A. An engineer wishing to deploy model 3B may examine these outliers to determine any correlation in image features. This may then allows for mitigation strategies at run-time or retraining with additional data samples.

Our method also allows for the generation of meaningful counter examples for image based classifiers. Figure 5 shows counterexamples for model 3A and illustrates the average level of haze which each class can withstand before misclassification occurs. This visual representation of perturbation levels allows domain experts to consider the robustness of the model with respect to normal operating conditions.

Having demonstrated our approach using the haze perturbation we now show results for the contrast and blur effects. Model accuracy in the presence of these perturbations is shown in Figure 6. We see that whilst the accuracy of models degrades as the amount of perturbation increases, the shape of the curves and the effect on individual models is different.

Model 3A was the most accurate model for much of the perturbation range under the effects of haze, while model 3B is superior with respect to contrast effects. We also see that while model 2B was relatively robust to haze, its robustness to contrast is poor. This shows that selecting a single model for all environmental conditions is unlikely to provide optimal performance. Our method allows for a greater understanding of models weaknesses when in the presence of natural phenomena and may allow for more intelligent choices to be made.

Fig. 5: Counterexamples for model 3A. Upper row is the original image, lower row has perturbation applied at the average level required for misclassification.



(a) Contrast (b) Blur

Fig. 6: Model accuracy with respect to increased contrast and blur effects

DeepCert with formal verification. For model 1A and 1B, we ran our method on the first 30 images correctly classified as class 3 in the test sets to compute the minimum values for hazing using contextual perturbations and for traditional l_1 norm perturbations. For all 30 samples the value of found through formal verification was the same as that for the test based verification, although we can not guarantee this to be true for all samples in the testing set.

Table 2 shows selected results from the formal verification compared with the test-based verification Sample #4 has an l_1 norm for model 1A that is lower than that of model 1B. This would indicate that model 1B is more robust. Examining contextual robustness, however, we see that model 1A is able to withstand more haze before misclassification occurs. A similar result is shown for sample #52. This time however model 1A would be judged more robust by the l_1 measure whilst model 1B is more robust according to the contextual measure. Other samples report identical l_1 measures between models (samples 114, 47, 3 and 15) yet their response to haze is different e.g. sample #114 using model 1A is able to withstand almost twice as much haze as model 1B.

These results demonstrate that our methods are able to use formal verification techniques, where the model form allows for such analysis. We also note that non-contextual point robustness is insufficient to assess the robustness of models in the presence of contextual perturbations.

Table 2: Minimum values for l_1 and hazing perturbation on test images.

	Model 1A			Model 1B		
	Verification		Test	Verification		Test
sample	l_∞	Haze	Haze	l_∞	Haze	Haze
4	0.002	0.623	0.623	0.006	0.525	0.525
114	0.002	0.451	0.451	0.002	0.225	0.225
47	0.006	0.592	0.592	0.006	0.752	0.752
52	0.006	0.830	0.830	0.010	0.654	0.654
3	0.010	0.764	0.764	0.010	0.713	0.713
15	0.010	0.760	0.760	0.010	0.810	0.810

Table 3: CIFAR-10 class descriptions

class	0	1	2	3	4	5	6	7	8	9
name	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

4.2 Case Study 2: CIFAR-10

In order to demonstrate that our approach is applicable to a range of problems we applied our method to a second well known classification problem, CIFAR-10. The data set consists of 60,000 32 32 colour images in 10 classes with 5000 training images and 1000 test images per class. Table 3 shows the names of the classes in this benchmark. The complexity and diversity of the images in this set is a more challenging classification task than the traffic sign problem. We again constructed models of increasing complexity with two models at each level. The accuracy of these models for the unperturbed test set is given in Table 4.

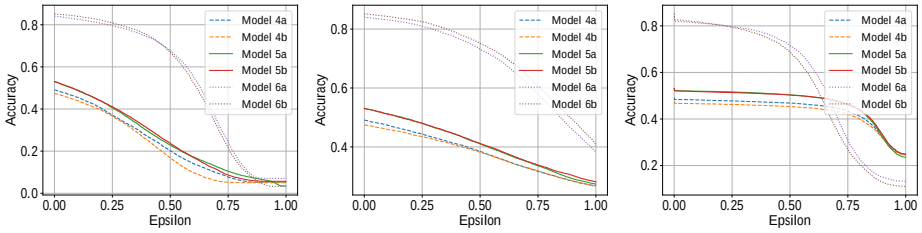
DeepCert with test-based verification. Model accuracy in the presence of the three forms of contextual perturbation are shown in Figure 7. We once more note the accuracy degrades as is increased for all perturbation types. For haze we observe a point at which the best model changes. This indicates that a system which is able to switch between models as the level of haze increases may demonstrate improved robustness. We also note that the CNN models outperform the simpler models by a significant margin under most conditions. For blur, however, when > 0.7 the CNN models under perform the simpler models.

Figure 8 shows the class accuracy for the CNN models subjected to the blur perturbation. We observe that the performance of classes between the models varies as shown in the traffic sign sign study. The accuracy of class 3 in model 6A, for example, is lower than that seen in Model 6B until > 0.7.

DeepCert with formal verification. Formal verification was applied to models 4A and 4B by again choosing 30 samples which we perturbed with haze. The results were in line with those found for the traffic sign model, but in addition we found a sample (#14) for model 4A which returned a lower robustness bound than when using test-based verification. Table 5 shows the predicted class $\hat{y}$ for this sample as is increased. We note that the sample is misclassified at $= 0.0723$ which was found using Marabou, it then returns to classifying the sample correctly before misclassifying again at $= 0.365$, the value found through testing. This confirms that, whilst testing may correctly identify the

Table 4: CIFAR-10 model accuracy

Model		Accuracy	Model		Accuracy	Model		Accuracy
4A	Small Relu	49.11	5A	Large Relu	53.20	6A	CNN	84.07
4B		47.45	5B		53.04	6B		85.17



(a) Haze (b) Contrast (c) Blur

Fig. 7: CIFAR-10 model robustness

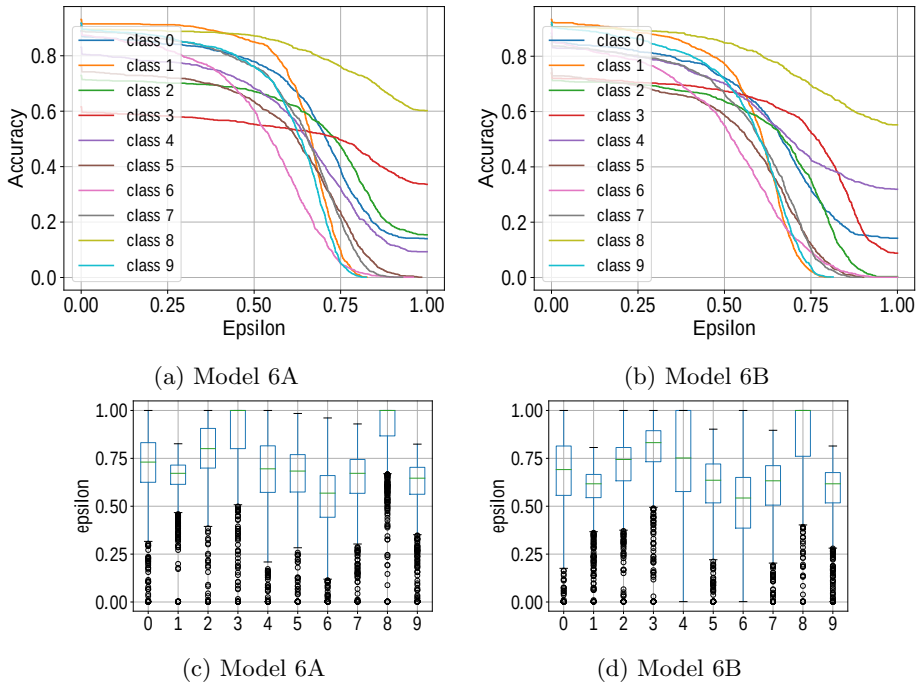
robustness bound for the majority of cases, formal verification is required for guarantees of robustness.

5 Related Work

It is well known [23] that neural networks, including highly trained and smooth networks, are vulnerable to adversarial perturbations; these are small changes to an input (which are imperceptible to the human eye) that lead to misclassifications. The vast majority of the work in this area focuses on formulating adversarial examples with respect to perturbations defined with L_p norms. The problem is typically formulated as follows: for a given network F and an input x, find an input x^0 for which $F(x^0) \neq F(x)$ while minimising $kx \quad x^0k$.

The metric used to compute the distance between points is typically the Euclidean distance (L_2 norm), the Manhattan distance (L_1 norm), or the Chebyshev distance (L_1 norm). Methods for finding adversarial examples and for checking robustness of neural networks to adversarial perturbations range from heuristic and optimisation-based techniques [8,14,18,2,17] to formal analysis techniques which are based on constraint solving, interval analysis or abstract interpretation [11,12,7,28,27,4,5]. In contrast to these works, which focus on local robustness, we take a more global view, as we aim to evaluate models on many input points and use the results to assess and compare models and inform developers' choices. Furthermore, we aim to study more natural (contextual) perturbations, as we do not limit ourselves to L_p norms.

Other researchers have started to look into robustness verification beyond the L_p-norm threat model. For instance, Semantify-NN [16] addresses robustness verification against *semantic* adversarial attacks, such as colour shifting and lighting adjustment. It works by inserting semantic perturbation layers to the input layer of a given model, and leverages existing L_p-norm based verification



(a) Model 6A (b) Model 6B

(c) Model 6A (d) Model 6B

Fig. 8: CIFAR-10 class robustness with respect to blur

Table 5: Formal versus test-based verification, correct label $y = 9$

	$\hat{y}$		$\hat{y}$
0.002	9	0.15	1
0.035	9	0.18	9
0.050	9	0.2	9
0.0723	1	0.03	9
0.1	1	0.365	2

tools to verify the model robustness against semantic perturbations. In our work, we also leverage an off-the-shelf verification tool (namely Marabou) to enable verification with respect to semantically meaningful perturbations. We do not modify the models, but instead encode the checks as Marabou queries.

6 Conclusions and Future Work

In this paper we have introduced DeepCert, a tool-supported method for the systematic verification of contextually relevant robustness for neural network classifiers. We have shown that the accuracy of a DNN image classifier is a function of the perturbation type to which sample images are exposed, and that through a systematic verification of the robustness with respect to these perturbations a more informed decision may be made to select a DNN model.

In future work we plan to investigate the use of alternative formal verification techniques with DeepCert, and the use of more complex models of natural

phenomena, parameterised for use within the framework. We also intend to investigate methods to allow for the systematic assessment of robustness within regions of the input space e.g. rain drops on a lens affecting part of an image.

Acknowledgements. This research has received funding from the Assuring Autonomy International Programme project 'Assurance of Deep-Learning AI Techniques' and the UKRI project EP/V026747/1 'Trustworthy Autonomous Systems Node in Resilience'.

References

1. Ashmore, R., Calinescu, R., Paterson, C.: Assuring the machine learning lifecycle: Desiderata, methods, and challenges. arXiv preprint arXiv:1905.04223 (2019)
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE symposium on security and privacy. pp. 39–57. IEEE (2017)
3. De Fauw, J., Ledsam, J.R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D., et al.: Clinically applicable deep learning for diagnosis and referral in retinal disease. Nature medicine **24**(9), 1342–1350 (2018)
4. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks. In: NASA Formal Methods Symposium. pp. 121–138. Springer (2018)
5. Fischetti, M., Jo, J.: Deep Neural Networks as 0-1 mixed integer linear programs: A feasibility study. arXiv preprint arXiv:1712.06174 (2017)
6. Gauerhof, L., Hawkins, R., Picardi, C., Paterson, C., Hagiwara, Y., Habli, I.: Assuring the safety of machine learning for pedestrian detection at crossings. In: International Conference on Computer Safety, Reliability, and Security. pp. 197–212. Springer (2020)
7. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.T.: AI2: safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy. pp. 3–18 (2018)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
9. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.: On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1503.02531 (2017)
10. Hamdi, A., Ghanem, B.: Towards analyzing semantic robustness of deep neural networks. In: European Conference on Computer Vision. pp. 22–38. Springer (2020)
11. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: International conference on computer aided verification. pp. 3–29. Springer (2017)
12. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: International Conference on Computer Aided Verification. pp. 97–117. Springer (2017)
13. 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)
14. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)

15. Mitani, A., Huang, A., Venugopalan, S., Corrado, G.S., Peng, L., Webster, D.R., Hammel, N., Liu, Y., Varadarajan, A.V.: Detection of anaemia from retinal fundus images via deep learning. Nature Biomedical Engineering **4**(1), 18–27 (2020)
16. Mohapatra, J., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: Towards verifying robustness of neural networks against a family of semantic perturbations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 244–252 (2020)
17. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
18. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy. pp. 372–387. IEEE (2016)
19. Picardi, C., Paterson, C., Hawkins, R.D., Calinescu, R., Habli, I.: Assurance argument patterns and processes for machine learning in safety-related systems. In: Workshop on Artificial Intelligence Safety. pp. 23–30 (2020)
20. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: CAV. pp. 243–257 (2010)
21. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Programming Languages **3** (Jan 2019)
22. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 international joint conference on neural networks. pp. 1453–1460. IEEE (2011)
23. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv:1312.6199 (2013)
24. Tabernik, D., Skočaj, D.: Deep learning for large-scale traffic-sign detection and recognition. IEEE Transactions on Intelligent Transportation Systems **21**(4), 1427–1440 (2019)
25. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th international conference on software engineering. pp. 303–314 (2018)
26. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356 (2017)
27. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (2018)
28. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th USENIX Security Symp. (2018)
29. Wu, H., Ozdemir, A., Zeljić, A., Julian, K., Irfan, A., Gopinath, D., Fouladi, S., Katz, G., Pasareanu, C., Barrett, C.: Parallelization techniques for verifying neural networks. In: 2020 Formal Methods in Computer Aided Design. pp. 128–137 (2020)
30. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: Attacks and defenses for deep learning. IEEE Transactions Neural Networks and Learning Systems **30**(9), 2805–2824 (2019)
31. Zhang, M., Zhang, Y., Zhang, L., Liu, C., Khurshid, S.: DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering. pp. 132–142. IEEE (2018)
32. Zhang, N., Zhang, L., Cheng, Z.: Towards simulating foggy and hazy images and evaluating their authenticity. In: International Conference on Neural Information Processing. pp. 405–415. Springer (2017)