Quantifying DNN Model Robustness to the Real-World Threats

Zhenyu Zhong, Zhisheng Hu, Xiaowei Chen
Baidu Security
{edwardzhong, zhishenghu, xiaoweichen01}@baidu.com

***Abstract*—DNN models have suffered from adversarial example attacks, which lead to inconsistent prediction results. As opposed to the gradient-based attack, which assumes white-box access to the model by the attacker, we focus on more realistic input perturbations from the real-world and their actual impact on the model robustness without any presence of the attackers. In this work, we promote a standardized framework to quantify the robustness against real-world threats. It is composed of a set of safety properties associated with common violations, a group of metrics to measure the minimal perturbation that causes the offense, and various criteria that reflect different aspects of the model robustness. By revealing comparison results through this framework among 13 pre-trained ImageNet classifiers, three state-of-the-art object detectors, and three cloud-based content moderators, we deliver the status quo of the real-world model robustness. Beyond that, we provide robustness benchmarking datasets for the community.**

***Keywords*-neural network, adversarial example, robustness, threat severity**

I. Introduction

Along came with the adversarial examples, attacking DNN models can be achieved through iterations of optimizations on the input using the gradient descent methods in the white-box setting [1] [2] [3] [4]. However, without a clear monetizing scheme on the adversarial examples, the attackers are less likely to perform a large-scale attack campaign in the real-world. On the other hand, DNN models also experience a hard time making consistent predictions in rare real-world scenarios, especially for those safety-critical applications, including autonomous driving, online content moderation. An occurrence of safety violation that could lead to severe consequences. More importantly, such a threat will not cease to exist even if there are no actual attackers. We regard the latter as the first and foremost threat to the reliability of the DNN in practice.

For understanding the real threat against the DNN models from various input perturbations from the real world, there are mainly two different approaches. One is collecting data from the real world [5], and the other is through data augmentation. The former one is cost-inefficient, and might not cover all the corner occasions; however, the latter one is widely used by the machine learning community with the affordable expense of GPU/CPU computations. The model built on these augmented data can further be adapted to scenarios with data scarcity issues by transfer learning [6] [7] [8].

In this paper, we go with the data augmentation to simulate the real world perturbations. We put these perturbations into five different safety property categories: *1) Luminance*: e.g., a glaring light could blind the camera and hurt the perception. *2) Spatial Transformation*: e.g., a loose camera might take in the data with a shifted angle. *3) Blur*: e.g., a fast-moving object is obscured when being captured by a slow camera. *4) Corruption*: e.g., possible malfunctioning camera sensor cells. *5) Weather*: e.g., invisibility caused by severe weather. Unlike the manual settings on the threat severity [9], and white-box based threat measurement [10] assuming the presence of an attacker, we extend the foolbox [11] and define the realistic threat severity as the real world minimal perturbations min_{pert} applied to the original input in order to change the model prediction behavior. Such threat severity is quantified in $||min_{pert}||_p$. The larger the $||min_{pert}||_p$, the less severe the threat is. That means it would require extra efforts to perturb the input to fool the underlying model. Terminology-wise, $||min_{pert}||_p$ also stands for the robustness of the model given an input. We interchange it with threat severity throughout the rest of the paper.

Furthermore, $||min_{pert}||_p$ can be different if the perturbed input is tested under different criteria. For example, a common misclassification on the perturbed input is relatively less severe than the targeted-class misclassification. In different to [11], we support measurement on different learning tasks.

In summary, we make the following contributions in this paper:

- We propose a set of safety properties observed from the real-world that could mislead the model prediction behavior.
- We extend the foolbox to search for the minimal real-world perturbations for each safety property on different learning tasks, especially for object detection.
- We demonstrate the realistic threats to near production-level DNN models across various learning tasks, including 13 ImageNet [12] scale pre-trained classifiers, three state-of-the-art object detectors, and three cloud-based content moderators.
- We provide benchmarking datasets for image clas-

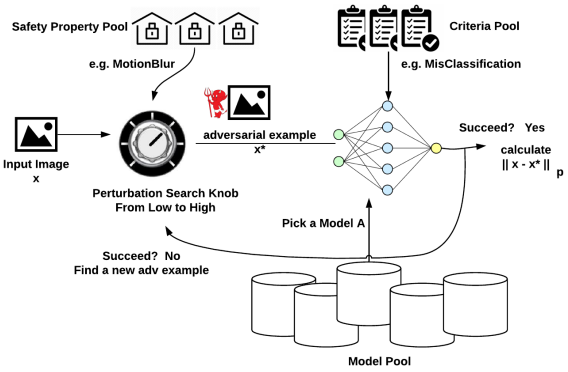


Fig. 1: Model Robustness Quantification Framework

sifiers, object detectors, and online content moderators from the realistic threat perspectives to the research community.

II. Approach

A. Framework

Figure 1 illustrates our model robustness evaluation framework. It consists of four components. *1) Safety Property Pool*, which contains 15 safety properties. *2) Threat Criteria* (*TC*), which define various aspects of the model robustness. *3) Model Pool*, which stores the target models whose robustness is measured. *4) Perturbation knob (PertKnob)*, which searches for the minimal amount of changes applied to the original image to mislead the prediction results. The entire robustness evaluation works as follows: Given an input image, we retrieve a safety property from the Safety Property Pool. The *PertKnob* incrementally increases the perturbations applied to the original input. The generated example is forwarded to the model. Based on the evaluation criteria chosen from the *TC*, only the one that successfully deceives the model will be saved, and the L_p norm-based distance is measured to indicate the model robustness. If the model makes a consistent prediction, *PertKnob* will increase the amount of the perturbation. The process stops either when it finds an example that deceives the model, or when it exhausts the perturbation search space without success.

B. Safety Properties

First of all, we define a set of safety properties observed in the real world. We divide them into five categories: *Luminance, Spatial Transformation, Blur, Corruption, and Weather*. Rather than increasing the cost of collecting data to cover rare cases, we leverage on computer vision simulations to achieve data augmentation. We discuss each of them in detail as follows.

1) Luminance:

a) Brightness: Changing the brightness of an image can be achieved by increasing or decreasing each pixel of the image by a constant value. *Clip* is a function to make sure the resulted new image x' is in a valid range of [0, 255] or [0, 1] typically.

$$x' = Clip(x \pm \epsilon) \tag{1}$$

b) Contrast Reduction: Usually multiplication and addition are used to adjust contrast and brightness [13]. We define ϵ as the contrast level and make it also have an impact on the constant factor C of the addition part.

$$x' = Clip((1-\epsilon)\cdot x + \epsilon \cdot C) \tag{2}$$

2) Transformation:

a) Rotation: Starting from the center point, the input image rotates in either $+180^\circ$ or -180° directions. We leverages on OpenCV API [14] to achieve this in Eq 3 in 2 steps. First, we supply c_x, the center of input x, and rotation factor ϵ to get the matrix after the rotation. Then based on the resulted matrix and the input x, we obtain the output x'.

$$\begin{aligned} m_{rotation} &= getRotationMatrix2D(c_x, \epsilon \cdot 180/pi, 1) \\ x' &= Clip(warpAffine(x, m_{rotation}, (w_x, h_x))) \end{aligned} \tag{3}$$

b) Horizontal/Vertical Translation: There are two directions a horizontal translation can happen: left and right, and for vertical translation: up and down. Similar to *Rotation*, we need to compute the matrix after we apply the amount of ϵ in either direction to the translation. Then we generate an image by calling $warpAffine$ function.

c) Spatial: This transformation combines rotation and translation to make it a more complicated case, such that the rotation does not have to happen from the center of the image. Correspondingly there are two factors, $\epsilon_{rotation}$ and $\epsilon_{translation}$, controlling the degree of the rotation and translation applied to the original image.

3) Blur:

a) Motion Blur: Applying motion blur is to have a function *CF* convolve a filter, a.k.a. kernel, across the image. As described in Eq 4, we fix the motion angle *ma* to be vertical v, f is the function generates the kernel. We only allow the kernel dimension ϵ_{dim} to determines the amount of blurring effects.

$$x' = Clip(CF(x, -1, k)), k = f(\epsilon_{dim}, ma_v) \tag{4}$$

b) Gaussian Blur: Similar to motion blur, applying gaussian blur is convolving a filter using a gaussian function G. The Gaussian distribution is determined by the standard deviation σ. We define ϵ_σ to control the blurring effects.

$$x' = Clip(G(x, \epsilon_\sigma)), G(x, \epsilon_\sigma) = \frac{1}{\sqrt{2\pi\epsilon_\sigma^2}} e^{-\frac{x^2}{2\epsilon_\sigma^2}} \tag{5}$$

4) Corruption:

a) Uniform Noise: Noise is generated separately based on a uniform distribution given the input range controlled by ϵ and is then added to the original input.

$$x' = Clip(x + \epsilon \cdot \mathcal{U}(\epsilon \cdot C)) \tag{6}$$

b) Gaussian Noise: Noise is generated separately from a normal distribution $\mathcal{N}$ with $\mu = 0$, std controlled by ϵ, and is then added back to the original image.

$$x' = Clip(x + \epsilon \cdot \mathcal{N}(\mu, \epsilon \cdot C)) \tag{7}$$

c) Blended Uniform Noise: Applying blended noise is to produce a randomized image x^* from a uniform distribution, such that the generated random image is sufficient to deceive the model. Then it is added to the original image, where we use ϵ to balance how much the resulting image comes from the random image, and how much comes from the original one.

$$x' = Clip((1-\epsilon)x + \epsilon \cdot x^*), x^* = \mathcal{U}(x_{dim}) \tag{8}$$

d) Salt & Pepper Noise: This type of noise emulates the malfunctions of the camera's sensor cell by uniformly flip the pixel to either white (pixel value 255) or black (pixel value 0). We use ϵ to control the amount of salt s and pepper p applied to the original image.

$$\begin{aligned} u &= \mathcal{U}(x_{dim}) \\ s &= (u >= 1 - \epsilon/2) \\ p &= -(u < \epsilon/2) \\ x' &= Clip(x + s + p) \end{aligned} \tag{9}$$

5) Weather:

a) Fog: A diamond square algorithm [15] is employed to emulate fog. It starts with a 2-dimensional map in the size of $(2^n + 1, 2^n + 1)$, with four corners initialized. Within that map, we can find both square shapes and diamond shapes. We run iterations to fill up the map. For each iteration, we set the center point of either a square or a diamond to the mean value of the corresponding corners plus a random variable W. We attempt various ϵ for W to control the output of the diamond square algorithm. As indicated in Eq 10, we also add a coefficient κ to the return value of the diamond square (DS) algorithm, and a coefficient γ_w to the parameter of DS that determines the W.

$$x' = Clip(x + \epsilon \cdot \kappa \cdot DS(\epsilon \cdot \gamma_w)) \tag{10}$$

b) Snow: We leverage on motion blur technique to create a snow filter in the size of one-hundredth of the original image. Then we use ϵ to determine the sparsity of the locations to convolve the filter over the original image. The volume of the snow is implicitly done by the density of the filters applied.

TABLE I: Threat Severity Criteria: x is the original input, $x + \delta$ is the perturbed input. C is the function returns class label, G is the ground truth of the input. P is the probability of the input prediction, $c = G(x)$, L is the multiclass label collection. D is the object detection function that returns a set of objects o. tc is the target class. † is the criteria related to object detection task.

Criteria	Description
Misclassification	$C(x+\delta) \neq G(x)$
ConfidenceMisclassification	$P_{\exists l \mid l \in \{L\}, l \neq G(x)}(x) \geq threshold_l$
TopKClsMisclassification	$C(x+\delta) \notin \{C_{topK}(x)\}$
Original Class Confidence Loss	$P_c(x+\delta) < threshold_c$
TargetClsProbBoost†	$P_{tc}(x+\delta) \geq threshold_{tc}$
TargetClsMisclassification†	$o_{tc} \in \{o : D(x)\}$, $o_{tc} \notin \{o : D(x+\delta)\}$

c) Frost: We create a frost template image T_{frost} and add it to the original image x. The portion of T_{frost} and x is determined by fixed-value parameters p_{frost} and p_x, as well as a variable ϵ.

$$x' = Clip(x \cdot \epsilon \cdot p_x + T_{frost} \cdot \epsilon \cdot p_{frost}) \tag{11}$$

C. Threat Severity Criteria

The threat severity of the violations to the safety properties mentioned above depends on various aspects of the model robustness. For the cloud-based content moderator, diminishing the NSFW confidence below the threshold would be a severe problem that an inappropriate image can be considered safe for the people under the age of 18, while for object detector, a target class misclassification would have more severe consequences. We refer to this as threat severity criteria. Table I gives a list of the threat severity criteria supported by our framework. It provides the flexibility to quantify the threat in various aspects of the model robustness.

D. Robustness Metric in L_p Norm

We measure the model robustness in Eq. 12, where m is the given model, c is the criterion used, k is the safety property under evaluation, R is the robustness. x_i is the input image randomly sampled from the dataset, x_i^* is the perturbed image, and N is the total number of images sampled. The minimal perturbation on a given image against model m is calculated in any L_p norm to the user's choice. The input image and the perturbation are normalized on a scale of 0 to 1, so is the L_p norm distance.

$$R_{m,c,k} = \frac{\sum_{i=1}^{N} Min_{m,c,k}(||x_i - x_i^*||_p)}{N} \tag{12}$$

E. Perturbation Knob

In our framework, *Perturbation Knob* is an important component to adjust the amount of perturbations during the model robustness evaluation. Throughout the Eq. 1 to 11, we define an ϵ to influence the outcome of the perturbation. Then we can apply a knob to search

in linear space for ϵ. Specifically, such linear space is defined within the range [0, 1]. We divide it equally into k cells, which specifies the granularity of the probes. The finer the granularity, the tighter the robustness, the more accurate the result we can get. To achieve a fine-grained robustness estimate and reduce unnecessary search cost, we set the k to 1000 in our experiment most of the time.

III. Experiment

A. Model Robustness in L_p

We conduct a comprehensive study on the model robustness against safety violations across various models of different learning tasks, including image classifiers, object detectors, and cloud-based content moderation models. We use L_2 distance throughout our experiments. The perceptual difference among various magnitudes of L_2 can also be found in Fig 10.

1) ***Image Classifier****:* We randomly sample $1k$ images from the ImageNet dataset. We choose $Misclassification$ from the criteria pool. As long as the generated perturbation is small enough to make the prediction label different from the ground truth, the resulting sample violates the corresponding safety property. We run all the $1k$ images per each safety property to get the min_{pert} in L_2 distance. The robustness is calculated based on Eq. 12. Fig 2 shows comparison results of 5 out of 15 safety properties over 13 pre-trained image classifiers. Among all these models, *Densenet* and *Resnet* provide the highest robustness most of the time. *Squeezenet* and *Alexnet*, on the contrary, are easy to break by a small amount of perturbation. We also detail out all the 11 pre-trained models we evaluate in Table II. The results for *Resnet*18, and *Resnet*34 are omitted intentionally because the results of $Resnet(50, 101, 152)$ are sufficient enough. In summary, all the existing image classifiers are susceptible to a small amount of perturbations.

We also provide the fooling success rate in Fig 3 for these five safety properties. We set the mean minimal L_2 distance as the threshold T for each property across all the models. We regard success if an input image needs less than T perturbation to achieve model misbehavior. In our result, *Densenet* and *Resnet* consistently outperform other model architectures with a fooling rate at 38.1% for *Brightness* test, while that for *squeezenet* can be as high as 91.3% for *Salt&Pepper* test.

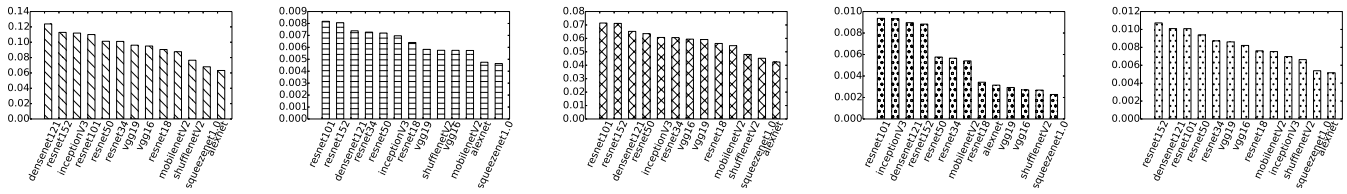
It is interesting to find that *squeezenet* with the fewest number of parameters has the worst robustness, while *alexnet* with 60 million parameters consistently ranks at the bottom over most of the safety violation tests. We suspect it is because *alexnet* employs fully connected layers, which results in an excessive number of parameters and is prone to overfitting. Thus the architecture extremely compact or composed of fully connected layers probably is not recommended for safe-critical applications.

Based on our result, the robustness does not have a clear correlation with the model complexity across different neural net structures. A large number of parameters do not necessarily lead to a more robust model. When considering structures within a family such as *Resnet*, higher complexity seems to provide more robustness. A *Resnet*-based architecture, including *Densenet*, which has a similar structure with deeper layers that concatenate previous layers, seems to be outstanding among all the structures. Thus it seems to be a preferred structure for safety-critical applications. However, the success of a neural network is also highly related to the activation function chosen, dropout strategies applied, and optimization techniques adopted. We will study how these contribute to the robustness of the model in our future work.

2) ***Object Detector****:* In this paper, we compare the following object detectors: a) $YOLOv3$ [16], which is one of the state-of-the-art object detectors. Its high throughput makes it suitable for real-time safety-critical applications such as autonomous driving. b) $SingleShotDetector(SSD)$ [17], which is as accurate as $YOLOv3$ but runs three times slower. c) *RetinaResnet* [18] usually has the best accuracy; however, it takes 198ms to complete an inference in order to get an accurate result.

We design our experiment as follows: In total, 849 images are selected. Each image contains an object that is significantly large so that all three object detectors predict that object the same as the ground truth with high confidence. We do that simply because many images from the MSCOCO dataset [19] contain very tiny objects where a misdetection on those tiny objects might not do severe harm. However, if an apparent large object is misdetected or misclassified, it would lead to severe consequences. Thus we only allow the following images to be selected:*1) The image contains the objects of the preferred class; 2) That object of the preferred class shall have an area at least as large as 5% of the entire image*. For simplicity, we set the preferred class to $'bus'$, such that all the object detectors agree on the ground truth of all the selected images. The model robustness is measured based on $Targeted\ Misclassification$ criterion such that the object is no longer predicted as $'bus'$.

We measure the robustness for all 15 safety properties for all three object detectors. To compare it with the image classifier, we also add *Resnet*152 for comparison. The result is presented in Figure 6. As we expect, object detectors behave more robust than the image classifier across various safety properties most of the time. In the best case, it takes 6.4 times more perturbations over a safety property named "Additive Uniform Noise" to fool YOLOv3 comparing to *Resnet*152, which is one of the most robust pre-trained image classifiers. Figure 4 demonstrates the fooling success rate with the mean L_2 distance for each property. We find that for *Brightness*, *RetinaResnet* has a higher fooling rate than



(a) Brightness (b) Gaussian Blur (c) Rotation (d) Salt and Pepper (e) Snow

Fig. 2: Model Robustness Ranking for the selected safety properties from each category (ImageNet dataset). Y-axis: Minimal perturbation in L_2

TABLE II: Robustness of 13 Pre-trained ImageNet models with input size (224×224): measured by the minimal perturbation in L_2 distance. L.→ Luminance, B.→ Blur, C.→ Corruption Noise, T.→ Transformation, W.→ Weather. For each safety property, the model with the highest robustness is highlighted. *Param* indicates the model size.

		squeeze net1.0	shuffle netV2	mobile netV2	dense net121	resnet 50	inception V3	resnet 101	resnet 152	alexnet	vgg16	vgg19
	Params ($\times 10^6$)	1.2	2.2	3.5	7.9	25.5	27.1	44.5	60.1	61.1	138.3	143.6
L.	Brightness	6.79e-2	7.65e-2	8.76e-2	**1.23e-1**	1.01e-1	1.11e-1	1.10e-1	1.12e-1	6.32e-2	9.50e-2	9.61e-2
	Contrast	2.84e-2	3.52e-2	4.0e-2	**6.29e-2**	4.63e-2	5.16e-2	4.99e-2	5.12e-2	2.57e-2	4.28e-2	4.39e-2
B.	Motion	7.73e-3	8.92e-3	9.07e-3	9.68e-3	9.49e-3	9.34e-3	9.91e-3	**1.00e-2**	8.21-e3	8.64e-3	8.84e-3
	Gaussian	4.64e-3	5.75e-3	5.73e-3	7.38e-3	7.19e-3	6.96e-3	**8.18e-3**	8.05e-3	4.75e-3	5.74e-3	5.83e-3
C.	Uniform	2.86e-3	3.08e-3	4.54e-3	1.21e-2	9.62e-2	1.22e-2	1.20e-2	**1.23e-2**	4.53e-3	4.86e-3	6.05e-3
	Gaussian	2.86e-3	3.03e-3	4.50e-3	1.17e-2	9.55e-3	1.20e-2	1.18e-2	**1.23e-2**	4.50e-3	4.80e-3	5.97e-3
	SaltPepper	2.28e-3	2.68e-3	5.40e-3	8.96e-3	5.76e-3	9.32e-3	**9.36e-3**	8.83e-3	3.15e-3	2.70e-3	2.93e-3
	Blended	7.32e-3	8.40e-3	1.11e-2	2.36e-2	1.86e-2	**2.59e-2**	2.21e-2	2.38e-2	8.86e-3	1.20e-2	1.38e-2
T.	Rotation	4.51e-2	4.80e-2	5.54e-2	6.52e-2	6.35e-2	6.05e-2	**7.14e-2**	7.11e-2	4.24e-2	5.94e-2	5.91e-2
	Horizontal	1.01e-1	1.06e-1	1.17e-1	**1.41e-1**	1.31e-1	1.19e-1	1.40e-1	1.43e-1	9.07e-2	1.31e-1	1.34e-1
	Vertical	9.59e-2	1.00e-1	1.13e-1	1.34e-1	1.27e-1	1.15e-1	1.36e-1	**1.37e-1**	8.37e-2	1.23e-1	1.24e-1
	Spatial	4.63e-2	4.91e-2	5.60e-2	6.75e-2	6.49e-2	6.19e-2	**7.30e-2**	7.25e-2	4.27e-2	6.02e-2	6.01e-2
W.	Fog	3.77e-2	3.83e-2	4.04e-2	4.73e-2	4.30e-2	4.27e-2	4.37e-2	4.49e-2	3.64e-2	4.47e-2	**4.54e-2**
	Snow	5.38e-3	6.63e-3	7.52e-3	1.00e-2	9.39e-3	6.96e-3	1.00e-2	**1.07e-2**	5.16e-3	8.21e-3	8.60e-3
	Frost	2.62e-2	2.93e-2	2.90e-2	3.42e-2	3.21e-2	3.32e-2	3.42e-2	**3.48e-2**	2.68e-2	2.95e-2	3.01e-2

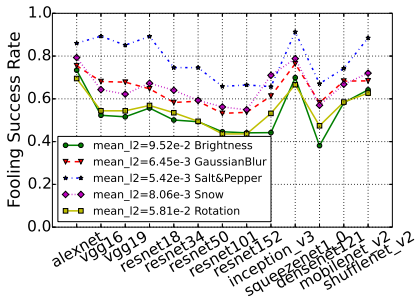


Fig. 3: Image Classifier

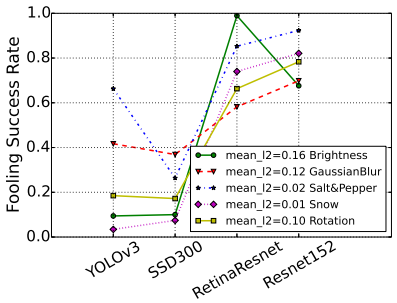


Fig. 4: Object Detector

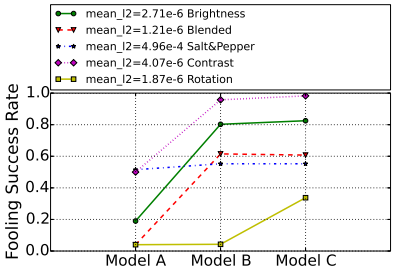


Fig. 5: Content Moderator

*Resnet*152. We suspect that it is due to the *Resnet*50 used by *RetinaResnet* as the backbone. According to Table II, *Resnet*152 is roughly 2.5 times more complex than *resnet*50, it is possible that in rare occasion for certain safety property, *Resnet*152 performs better than *RetinaResnet*(50).

a) Resilience to Weather Changes: We do not show fooling success rate for weather-related safety properties in Figure 4, mainly because we find object detectors are surprisingly robust to those changes. According to Table III, *YOLOv*3 has the best resilience to *Snow*&*Frost*, and *SSD*300 has the best resilience to *Fog*. The high

TABLE III: Percentage of images preserve their ground truth labels after exhausting the search space.

	YOLOv3	*SSD*300	*RetinaResnet*
Fog	0%	99.76%	76.9%
Snow	88.9%	45.2%	2.9%
Frost	59.5%	26.7%	4.8%

percentage in the table means the portion of the images preserves the original labels after the *PertKnob* exhausts the search space. The minimal perturbations measured in Figure 6 for weather-related safety properties are the means over the samples that do lead to mispredictions.

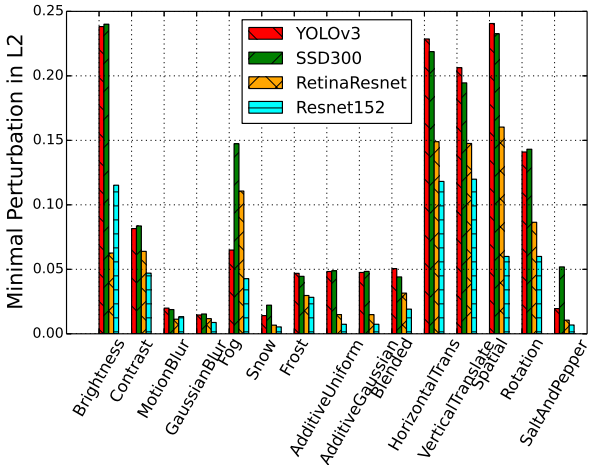


Fig. 6: Robustness Comparison among *YOLOv*3, *SSD*300, *RetinaResnet* and *Resnet*152 with input size (416×416).

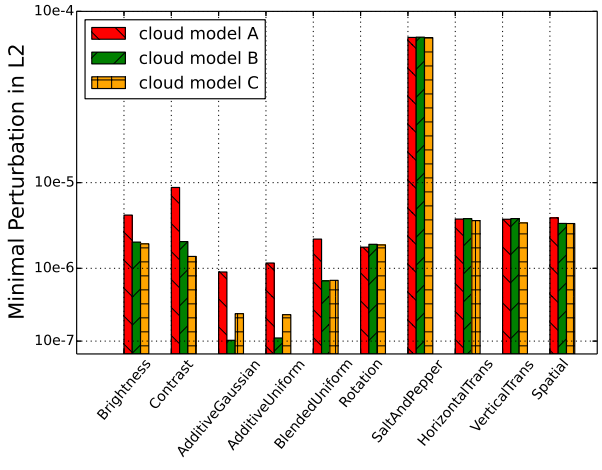


Fig. 7: Robustness Comparison among 3 commercial content moderators with input size (224×224)

b) Label Choice Impact on the Robustness: The robustness varies with different target object classes since the robustness sensitivity of different classes is not the same. If we choose 'person' instead of 'bus', the robustness threshold for comparison will be different. For the 'Misclassification' criterion used for image classifier evaluation, the robustness threshold remains the same.

3) ***Cloud-based Content Moderator****:* Machine learning as a service(*MLaaS*) offers API access to perform model inference on the given inputs. Content moderation is one of the important tasks to prevent any access to inappropriate content such as NSFW. We choose three commercial *MLaaS* providers and randomly selected 400 NSFW images [20] and ran through 10 different safety properties as indicated in x-axis label in Figure 7. Since the output format of the cloud-based models varies, the prediction result can either be the logits, or the class labels. Thus we employ a combination of $Misclassification$, $Original\ ConfidenceLoss$, and $TopKMisclassification$ for the test criteria. Our goal is to find minimal perturbation to the input so that it is no longer considered porn. We do not include *Blur* category in this test because the low quality from blurring the image is unlikely an incentive to the attackers. The results indicate that with a small amount of perturbations in the magnitude from 10^{-7} to 10^{-4}, these cloud-based content moderation models are easy to break. Interestingly, all three models have very close robustness for safety properties related to spatial transformation. $Model_A$ is slightly better than the other two on *Luminance* and *Corruption* related properties with an exception that all three models perform roughly equal robust against $SaltAndPepperNoise$.

B. Property-specific Robustness Metrics

Though L_p norm-based metric is widely used for scientific research, it is difficult to comprehend from normal users' perspectives. We find some alternative metrics, which are easier to explain for some safety properties.

1) Rotation Angles: In our experiment, we rotate the image from $-180°$ to $+180°$. It would make more sense to use the ranges of angles as the amount of perturbations applied to the original image. Figure 8 demonstrates the statistics of the rotation angles needed to fool the models from different learning tasks. We only compare the results within each learning task because of the different datasets used in the experiment.

For each input image, we start with $0°$ and rotate until the image is misclassified. The upper and lower bound of the box indicates the mean of the minimal angle the image needs to rotate towards both $+180°$ and $-180°$ over a randomly sampled 1000 images. The upper and lower ceilings indicate the largest minimal angle (the worst case), in which the image needs to rotate towards either direction. The result consistently confirms our previous findings that *Resnet*152, *Densenet*121 tolerate more image rotation, while *alexnet* has the least toleration. Similarly, *ssd*300 performs the best for object detector. All of the cloud-based models can make consistent porn detection with a broader range of rotation angles. Interestingly, the amount of rotations towards either direction seems near symmetric.

2) Kernel Dimension for Motion Blur: Since we leverage OpenCV [14] to implement Motion Blur property, the amount of the perturbation depends on a parameter called the kernel dimension, a.k.a. kernel size (*KS*). The larger the *KS* (perturbation) the model can tolerate, the more robust the model is. Thus the *KS* can be a supplementary metric for the model robustness evaluation. Figure 9 demonstrates the *KS* required to achieve enough motion blur perturbations on the input image. The solid points connected by a dotted line are the mean *KS* over the test dataset for a given model. The upper and lower

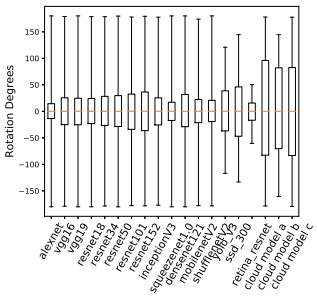


Fig. 8: Rotation Angle

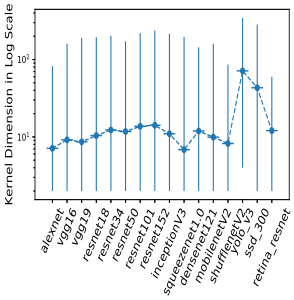


Fig. 9: Kernel Size

Fig. 10: Benchmarking Dataset: Images causing mispredictions with least perturbations.

bounds indicate the largest and smallest minimal *KS* ever reached by an input. The result proves that object detectors require larger *KS* to generate blurs to deceive the models.

C. Benchmarking Datasets

In this work, we produce benchmarking datasets [21] , including all the least perturbed examples generated across different types of models based on various safety properties. Fig 10 gives an example of this dataset. The leftmost column contains the original image from MSCOCO dataset. The images from the second to the rightmost column are generated against *YOLOv*3, *RetinaResnet*, and *Resnet*152 with corresponding L_2 distances in the magnitude of $10^{-1}, 10^{-2}, 10^{-3}$ respectively. Each row includes the perturbed examples that can successfully deceive the underlying models according to a specific safety property with *uniform noise* in row 1, *brightness* in row 2, *contrast* in row 3, and *rotation* in row 4. We leave out the examples generated from other safety properties due to the limited space. In summary, our benchmarking dataset contains all the safety-violation images including 192,591 images generated based on ImageNet, 52,207 images generated based on MSCOCO , and 13,690 images generated based on the NSFW.

IV. Related Work

Adversarial examples (AE) [1] [2] [22]can be crafted to confuse deep learning models in a systematic way assuming white-box access. Later on, researchers took physical conditions into account and came up with more advanced approaches [23] [24] [25]to confuse physical objects like stop-sign and vehicles. In practice, the model's parameters and structure are unavailable to the attacker. The adversary has to leverage on the transferability [26] [27] of AE. However, the chance of any large scale AE attack campaign in the real-world is slim.

Recent efforts [28] [29] [30] [31] [32] seek formal methods to derive provable robustness by leveraging techniques such as SMT solver and abstract interpretation. However, none of the existing tools can scale to practical datasets (e.g., ImageNet) and network structures. Huang, et al., [33] seem to be able to check the safety of deep neural networks for realistic images. Its efficiency suffers from the exponential increase in the number of features and the prohibitive complexity for larger images. Thus state-of-the-art formal methods are not easy to be applied in practice. Certified robustness via randomized smoothing [34] [28] derives the certified robustness in L_2 norm. Our method provides the flexibility for the users to choose any L_p norm to their needs.

Hendrycks et al. [9]'s benchmarking work is the closest one to ours. They studied model robustness to the common corruption. Rather than following their manual threat severity setup, we produce a tighter and more accurate robustness boundary for the models. ObjectNet [5] collects subtle data to confuse object detectors and shows a 40-50% drop on the performance. It can be combined with our approach to produce more data augmentation for robustness evaluation in a safety-critical context.

V. Conclusion

In this paper, we discuss the pressing need for DL model robustness evaluation in safety-critical settings with no presence of the attacker. We proposed a systematic framework to measure the minimal perturbations needed to achieve violations against safety properties. Our results from the evaluation of ImageNet classifiers, object detectors, and content moderators indicate the status quo robustness. The images introducing the violations are open to the community.

REFERENCES

[1] David Wagner Nicholas Carlini. Towards evaluating the robustness of neural network. In *Proceedings of the 38h IEEE Symposium on Security and Privacy*, 2017.

[2] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 387–402, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, page arXiv:1412.6572, Dec 2014.

[5] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9448–9458. Curran Associates, Inc., 2019.

[6] Jonathan Baxter. Learning to learn. chapter Theoretical Models of Learning to Learn, pages 71–94. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[7] Chuong B. Do and Andrew Y. Ng. Transfer learning for text classification. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 299–306, Cambridge, MA, USA, 2005. MIT Press.

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[9] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[10] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, Proceedings*, pages 644–661, 2018.

[11] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[13] Changing the contrast and brightness of an image. "https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html", 2018. [Online; accessed 13-Dec-2019].

[14] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[15] Alain Fournier, Don Fussell, and Loren Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, June 1982.

[16] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. 2016. To appear.

[18] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv e-prints*, page arXiv:1405.0312, May 2014.

[20] Nsfw dataset. "https://github.com/sajithm/nsfw-v1", 2018. [Online; accessed 13-Dec-2019].

[21] Baidu perceptron robustness benchmarking dataset. "https://github.com/advboxes/perceptron-benchmark/tree/master/dsn$_b$*enchmark*".

[22] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. *arXiv e-prints*, page arXiv:1511.07528, Nov 2015.

[23] Zhenyu Zhong, Yunhan Jia, Weilin Xu, and Tao Wei. Perception deception: Physical adversarial attack challenges and tactics for dnn-based object detection. *BlackHat Europe*, 2018.

[24] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 1989–2004, New York, NY, USA, 2019. ACM.

[26] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv e-prints*, page arXiv:1605.07277, May 2016.

[27] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.

[28] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

[29] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[30] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2266–2276. Curran Associates, Inc., 2017.

[31] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8400–8409. Curran Associates, Inc., 2018.

[32] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Towards proving the adversarial robustness of deep neural networks. *arXiv preprint arXiv:1709.02802*, 2017.

[33] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

[34] Jinyuan Jia, Xiaoyu Cao, Binghui Wang, and Neil Zhenqiang Gong. Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing. In *International Conference on Learning Representations*, 2020.