Assistive Signals for Deep Neural Network Classifiers

Camilo Pestana
camilo.pestanacardeno@research.uwa.edu.au

Wei Liu
wei.liu@uwa.edu.au

David Glance
david.glance@uwa.edu.au

Robyn Owens
robyn.owens@uwa.edu.au

Ajmal Mian
ajmal.mian@uwa.edu.au

The University of Western Australia
35 Stirling Hwy, Crawley WA 6009, Australia

Abstract

Deep Neural Networks are brittle in that small changes in the input can drastically affect their prediction outcome and confidence. Consequently, research in this area mainly focus on adversarial attacks and defenses. In this paper, we take an alternative stance and introduce the concept of ***Assistive Signals****, which are perturbations optimized to improve a model's confidence score regardless if it's under attack or not. We analyze some interesting properties of these assistive perturbations and extend the idea to optimize them in the 3D space simulating different lighting conditions and viewing angles. Experimental evaluations show that the assistive signals generated by our optimization method increase the accuracy and confidence of deep models more than those generated by conventional methods that work in the 2D space. 'Assistive Signals' also illustrate bias of ML models towards certain patterns in real-life objects.*

1. Introduction

The study of adversarial examples in computer vision is often motivated by using human indistinguishable small changes in an input image to cause model prediction errors [25, 11, 21, 9]. In this threat scenario, the adversary is allowed to perturb all pixels in an image such that the ℓ_p-norm of the perturbation is constrained to be within a prescribed bound. However, many of these attacks are shown to be rather inefficient in real-life scenarios [8, 19]. Adversarial patch attacks [5, 19, 17], where patches of certain patterns are added to input images, are one of the best attacks in the real-world.

Consequently, research in adversarial machine learning (ML) has been focused around creating efficient attacks, better defenses and exploring robustness in Deep Neural Networks (DNNs) [1, 10, 3, 27, 18, 23]. In this paper, we look at the problem on how a physical object or its image can be guaranteed to be correctly recognized despite adversarial conditions in the 3D scene. Perturbations on the input images or alterations in the object are used to improve the prediction confidence of the correct class through Assistive perturbations.

We coined the term *Assistive Signals* to refer to any embedded signal in the input that enhances the confidence score of the prediction made by a classifier at inference time to recover with 'good' confidence the correct class. For example, if a 3D object (e.g., car) is attacked by an adversary modifying scene properties such as object pose or lighting so that it is misclassified by a DNN, then 'Assistive Signals' embedded in the 3D object should ideally 'assist' the model to correctly classify the object despite the adversarial conditions. 'Assisting' a ML classifier is the main purpose of the *Assistive signals*, and such signals can be expressed in many forms including but not limited to *Assistive Textures* and *Assistive Patches*. Therefore, the term 'Assistive Signals' confers enough meaning to give an intuition about its purpose and is also sufficiently generic to include all future work in regards to the many types of assistive signals that could be created in this emerging area of research.

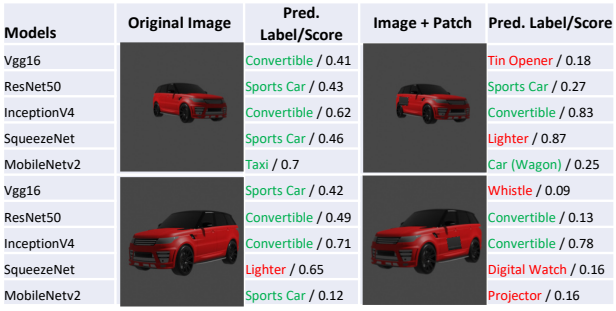

Figure 1: Arbitrarily added a non-adversarial black patch to a 3D mesh of a car and classified the rendered images from different angles. We considered as valid (in green) the predicted classes related to cars.

The concept of 'Assistive Signals' might seem similar to adversarial attacks in that both approaches can perturb or alter the input. It can also appear similar to defenses in that both could improve the classifier's confidence scores under attack. However, despite the similarities between these concepts, there are some key differences:

First, Assistive Signals are meant to improve the general ability of classifiers to recognize the target object predicting the correct class (in some cases with higher confidence). Most defenses focus in the recovery of the accuracy only when the classifier is under attack and the incorrect class is predicted while degrading their performance in 'normal' conditions.

Second, current literature in adversarial ML focus on the software/algorithmic side, where attack or defense tech-

niques are developed for. Instead, the Assistive Signals are operating in the physical space, i.e. how we alter the appearance of an object to protect it from being misclassified (e.g. a car).

Third, Assistive Signals make the object itself easier to classify. For example, if a self-driving manufacturer company improves the design of their cars to be more 'detectable' using the insights from Assistive Signals (i.e, Robust Patterns), they would be making their cars more recognizable in general, agnostic to any computer vision system improving road safety.

Our biggest motivation for the creation of Assistive Signals embedded into objects is the fact that fooling a model in the 3D scene is easier than making it robust, as demonstrated by Zeng et al. [26] and Alcorn et al. [2]. Hence, Assistive Signals are a more challenging task than creating adversarial signals. Figure 1 illustrates how even a simple non-adversarial black patch (i.e. occlusion) and changes in the 3D scene properties such as camera view and lighting can affect significantly the confidence scores of different ImageNet-based [15] models. Therefore, creating crafted adversarial attacks for physical objects is not necessary to fool a model when even simple changes in the environment pose a significant threat to the model. These challenges can be addressed using Assistive Signals by improving the object's design using the insights from *Robust Patterns*, or modifying the appearance of existing physical objects with Assistive Textures and Patches. The main contributions of this paper are thus three-fold:

- We propose the first 'adversarial' learning technique that optimizes perturbations in the input for the purpose of *improving* the confidence score of an image classification model instead of degrading its performance.
- We propose an algorithm[1] that can generate *Assistive* textures and patches that are invariant to 3D physical properties such as the camera orientation and lighting conditions etc.
- We introduce the concept of *Robust Patterns* that can be extracted from our Assistive Signals to illustrate bias in DNNs.

2. Problem Definition

Let $x \in \mathbb{R}^{H \times W \times 3}$ denote an image with height H, width W and three color channels (i.e, RGB). A perturbation ρ is quasi-imperceptible if $d(x, x+\rho) < \epsilon$, where d is a distance metric and ϵ is a small number. Let $y = \{p_1, \ldots, p_m\} \in R^n$ be the probabilities of the predicted labels 1,..., m. If $\mathcal{C}$ is an image classifier, then $\mathcal{C} : x \rightarrow \{k(x), p_k(x)\}$, where $k(x)$ is the label of x with the highest confidence $p_k(x)$. An **assistive image** is defined as $x_{asst} = x + \rho_{asst}$, such that if $\mathcal{C}(x) = \{k(x), p_k(x)\}$ and $\mathcal{C}(x_{asst}) = \{k(x_{asst}), p_k(x_{asst})\}$, then $k(x) = k(x_{asst})$ and $p_k(x_{asst}) > p_k(x)$. In other words, the classifier returns the same correct label with a higher confidence. In contrast, adversarial signals have the opposite aim, where $x_{adv} = x + \rho_{adv}$ return the incorrect label.

[1]Code available at https://github.com/elcronos/assistive-signals

Algorithm 1: Assistive Textures

Input: A render function $\mathcal{R}$, ensemble of classifiers $\mathcal{C}$, 3D mesh/scene parameters Θ, a mask $\mathcal{M}$, target label y_t, step size α, number of iterations γ
Output: UV/Vertex texture Θ_t
MaskedTexture $\mathcal{R}, \mathcal{C}, \Theta, y_t, \alpha, \gamma$
  $\xi \leftarrow 0$
  while $\xi \leq \gamma$ **do**
    $\Theta_t \leftarrow$ applyMask$(\Theta_t, \mathcal{M})$
    $\mathcal{I} \leftarrow \mathcal{R}(\Theta)$
    L = $\mathcal{L}(\mathcal{C}(\mathcal{I}), y_t)$
    $\Theta_t \leftarrow$ Clip$(\Theta_t + \alpha \nabla L, 0, 1)$
    $\xi \leftarrow \xi + 1$
  if has_mask$(\mathcal{M})$ **then**
    $\mathcal{P} \leftarrow$ getPatch$(\Theta_t, \mathcal{M})$
  else
    $\mathcal{P} \leftarrow$ None
  return Θ_t,$\mathcal{P}$

3. Assstive Signals Generation

Most adversarial attacks focus on how to digitally alter natural images [16, 20, 6, 22]. We focus in the alteration of 3D meshes as a proxy for real life, where we can simulate different camera-views and illumination conditions in the scene. In this section we present how Assistive Signals are generated using 3D mesh models (Alg. 1).

A rendering function $\mathcal{R}$ takes shape parameters Θ_s, camera parameters Θ_c, lighting parameters Θ_l and texture parameters Θ_t as inputs and outputs a batch of rendered RGB images $\mathcal{I}$. Let's define the inputs for $\mathcal{R}$ as $\Theta = \{\Theta_s, \Theta_c, \Theta_l, \Theta_t\}$. The function $\mathcal{R}$ computes the gradients of the output image with respect to the input parameters $\frac{\partial \mathcal{I}}{\partial \Theta}$. While inferring Θ_s and Θ_c from $\mathcal{I}$ is a common task using differentiable renders [14], to calculate our assistive textures, we will focus on inferring the parameter Θ_t. We assume the parameter Θ_t to be either a Vertex or UV Texture. In addition, $\mathcal{C}$ is a visual classifier. Let y be the true label and y_t the target. We use cross-entropy as the loss function $\mathcal{L}$ for image classification networks such as those trained on ImageNet. We use the Clip mathematical function to keep the RGB values of the texture in the range [0,1] or [0,255].

Algorithm 1 illustrates the general steps to generate an Assistive Texture Θ_t targeting the label y for the rendered views $\mathcal{I}$ calculated for a given 3D mesh object. Therefore, given all the parameters needed, the algorithm first generates images from a 3D mesh object using the rendering function $\mathcal{R}$ and parameters Θ, the number of images generated in this step corresponds to the number of cameras passed in Θ_c. The Assistive Texture algorithm applies the full/partial perturbations over the texture Θ_t, which can be a Vertex or UV texture. If a mask $\mathcal{M}$ is provided, the algorithm will apply a partial perturbations in the object to create *Masked Textures* or *Assistive '3D' Patches*. Masked Textures and Patches work similarly in practice, however, we differentiate the patches in that the perturbation is localized in a "relatively" small area of the 3D object (e.g. a car with a small square-shaped assistive patch on the side) whereas Masked Texture could cover the majority of the 'car' in this example, but with the exception of windows, front lights, etc. In the 3D space, we can apply a patch similarly to the approach by Brown et al. [6] only when dealing

with a UV Texture. In that case, an extra argument such as a 2D mask with the location of the patch is needed. On the other hand, for a Vertex Texture, every vertex in a mesh can optionally store an RGB color value. For this case, a mask can be a 1D vector for each binary values for each vertex.

4. Experimentation

In this section we perform experiments to test the effectiveness of Assistive Signals for different tasks such as improving the confidence score in 2D images w.r.t. their ground truth label, the 'Recognizability' of different 3D objects when using Assistive Textures and Patches in the 3D space, and explaining visual salience features that might contribute to make an object more recognizable to a deep learning classifier.

4.1. Assistive Patches

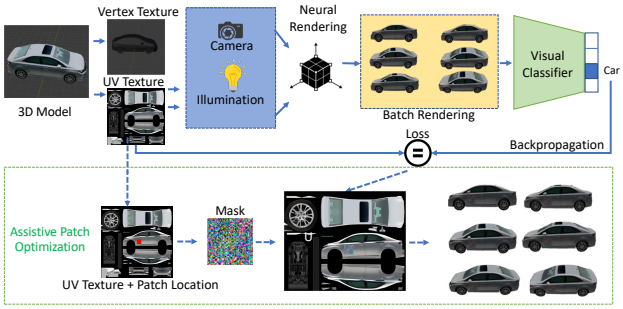

Figure 2: Example of Assistive Patch optimization using a 3D mesh of a car for a visual classifier (DenseNet121 here) following the steps from Algorithm 1.

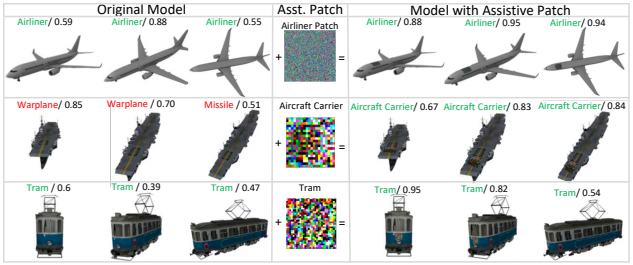

Figure 3: Assistive Patches created for 3 different objects. Each patch was optimized with differentiable rendering using batch rendering sizes of 15, which includes different views of the object and changes in illumination. The patches were optimized for ResNet50 [12].

Adversarial patches [6, 4] have been tested before in the physical world usually by printing the adversarial patterns and then taking pictures of the altered scene containing the patch. One characteristic that makes adversarial patches powerful is the ability of being universal (i.e. they can be applied to any image or scenery). In Table 1, we show the comparison between Assistive Patches using the traditional EoT framework from Brown et al. [6], which optimizes using transformations on images to make the patch location invariant and our approach optimizing the patches in 3D scene to be light/position invariant (See Fig. 2 and Alg. 1),

Table 1: Confidence scores (correct class) for four 3D models imaged from three different views where the illumination was also changed for each view (See Fig. 3). Using DenseNet121 [13], assistive patches created using conventional methods (optimized with 2D images) [6] fail or give low confidence for the correct class. However, 3D assistive patches (optimized with 3D neural rendering) always improve the confidence score of the correct class for all views. If the DNN is unable to predict the correct class an **x** mark is shown instead of the the confidence score.

	Original			**2D Assistive Patches**			**3D Assistive Patches**		
	Views			Views			Views		
3D Model	1	2	3	1	2	3	1	2	3
Airliner	0.59	0.88	0.55	0.57	0.89	0.60	0.88	0.95	0.94
Car	0.65	x	x	0.73	x	x	0.97	0.8	0.71
Aircraft Carrier	x	x	x	x	x	x	0.67	0.83	0.84
Tram	0.60	0.39	0.47	0.76	0.42	0.46	0.95	0.82	0.54

Table 2: Transferability of assistive textures generated from different deep models. Correct class confidence scores for a 3D model imaged from three different views where the illumination was also changed for each view are shown. Crosses mean the predicted class was incorrect. First row shows the original object. F_a is a full assistive texture with $\epsilon = 0.01$ and $n = 50$ iterations. M_1 a is masked assistive textures with perturbation $\epsilon = 0.01$ and $n = 10$ iterations. An **x** mark is shown when the given DNN predicts the incorrect class.

	Train/Test	ResNet50			DenseNet121			InceptionV3		
Views		1	2	3	1	2	3	1	2	3
	Original	x	x	x	x	x	x	x	0.87	0.33
F_a	ResNet50	0.99	0.98	0.99	0.4	0.74	0.7	x	0.6	x
	DenseNet121	0.13	0.37	0.6	0.99	0.99	0.99	x	0.84	x
	InceptionV3	x	0.29	0.09	x	x	x	0.99	0.99	0.99
M_1	ResNet50	0.99	0.99	0.99	0.33	x	0.21	0.6	0.99	0.69
	DenseNet121	x	x	x	0.99	0.99	0.99	0.96	0.99	0.84
	InceptionV3	x	x	x	x	x	x	0.99	0.99	0.99

which render views of the 3D object from multiple angles and optimizes the patch end-to-end from the rendered images to the 3D mesh object (i.e, differentiable rendering). In Table 1 we show that our approach is better than 2D Assistive Patches optimized using the traditional algorithm EoT. Usually, Patches optimized using 2D images fail when tested in the real-world. A more effective approach is simulating the 3D environment and train the patches under different light conditions, cameras, etc. like in our approach. Moreover, in Figure 3, we show different 3D objects and the Assistive Patches created for those objects. Once we apply an Assistive Patch, the confidence score with regards to the 3D object in the scene improves. This holds for different illumination conditions and camera viewpoints in the 3D scene.

4.2. Assistive Textures

Our method uses differentiable rendering, which allows to optimize the texture end-to-end from the rendered images to the 3D mesh object to make it location, lighting, and camera view-point invariant.

In the case of masked textures, we can leave out certain regions e.g. the windows, lights and tyres of a car. For instance, Figure 4 compares an unconstrained full body texture (row b) to *Masked Textures* (row c and d) on a 3D model from the ShapeNetCoreV2 [7] dataset. The masked

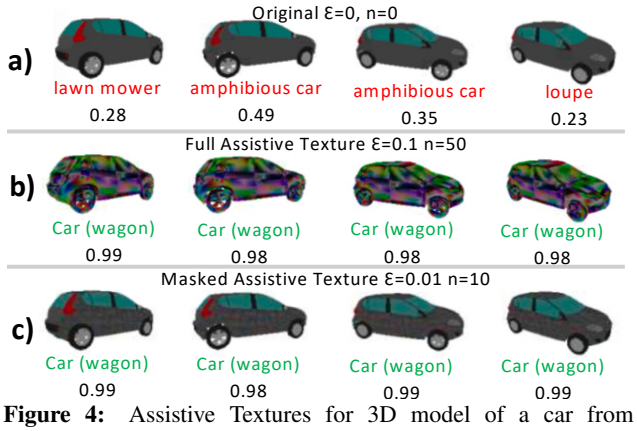

Figure 4: Assistive Textures for 3D model of a car from ShapeNetCoreV2 [7]:(a) The original object is misclassified by InceptionV4 [24]. (b) Full assistive texture. (c) Masked assistive texture after 10 iterations contrained with $\epsilon = 0.01$.

assistive textures have been optimized only on specific areas of the car as opposed to its full body perturbation. In the case of row (c), where the perturbation applied is quasi-unnoticeable if compared to the original model (a), the assistive signal improves significantly the right class 'car (wagon)'. An interesting point to note is that full textures tend to be smoother compared to masked textures which appear more 'patchy' or noisy. One possible explanation for this phenomenon is that by completely removing the texture from the 3D object (a car), we encourage the full texture optimization to find these 'Robust Patterns', which are high-level features that are not present in the original 3D object. In contrast, when creating a masked texture, the texture is partially removed leaving out important features (e.g, windows, tyres) and hence, the algorithm does not need to find those important features anymore and focuses on optimizing other features that look like noisy patterns similarly to conventional crafted perturbations. Moreover, we tested if the assistive textures optimized using differentible rendering for one deep model are transferable to other deep models. Results are shown in Table 2. We can observe that the full assistive texture F_a transfers reasonably well across different CNN-models. M_1 also transfers well across different DNNs and do not need a high magnitude of perturbation.

4.3. Robust Patterns

Patterns created in ℓ_p-unconstrained assistive textures can reveal visual salient patterns, which are human-readable and can be used to provide some insights into what patterns are more detectable to a model. This is what we refer to as *Robust Patterns*. Figure 5 illustrates an example of *robust pattern* that can be used as a method for explainability, but more importantly, it can also provide information about what patterns from the latent space of the trained model are more recognizable to a deep neural network. Full textures are highly transferable to other DNNs, this means that the visual salient patterns contains 'universal' shared abstractions of the target object that are understood by other DNNs. In some other experiments, we found that they can also provide insights into what colors are more detectable for an specific object. For instance, depending on the country the body of a traffic light can be sometimes gray (metallic), yellow or black. However, the robust patterns obtained from different ImageNet classifier models indicate that 'black' is the preferable color. This also gives intuitions into what sort of data was used for training an specific algorithm.

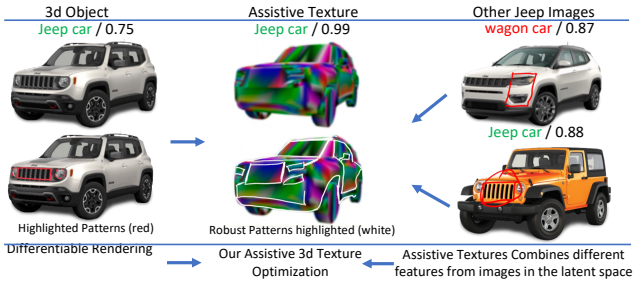

Figure 5: Robust Design example: First column shows an example of a 3D model (jeep). The second column highlights robust patterns (white lines). The original rendered image has rounded-shaped front lights. In the second column, after applying a full assistive texture from the original 3D object, we can observe that the visual salience patterns such as front lights and grilles changed significantly (e.g., having rectangle-shaped front lights instead of rounded and a different design for the grill).

In Figure 5, we found other images corresponding to different models of existing jeep cars outside the ImageNet dataset and we noticed the similarities of the patterns highlighted by the *robust patterns* with other jeep models. In the process of the texture optimization, the algorithm finds patterns from the latent space that if combined and applied to the texture it improves the confidence score of the targeted class. In the third column we highlighted the features from different jeep cars that are similar to the ones displayed in the assistive texture. Despite the original 3D model having rounded lights and very narrow grilles, in the optimization the assistive signal seems to combine different features from other jeep models to create a new design.

5. Conclusion

We introduced the concept of *Assistive Signals* which, as opposed to Adversarial Examples, aim to improve the confidence score of the ground truth class. At the same time, we provided the first comprehensive study about the efficacy of Assistive Signals simulated in a 3D environment and provide algorithms for the creation of textures and patches. Our approach re-purposes image perturbation for "good" going beyond deceptive signals with the aim of fooling a model. Future directions could extend outside the imaging boundaries and tackle similar problems in other type of data such as Point Clouds, Videos, Sound, etc.

6. Acknowledgements

The main author was recipient of an Australian Government Research Training Program (RTP) Scholarship at The University of Western Australia. This research was supported by ARC Discovery Grant DP190102443.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *arXiv preprint arXiv:1801.00553*, 2018.

[2] Michael A. Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects, 2019.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[4] A. Braunegg, Amartya Chakraborty, Michael Krumdick, Nicole Lape, Sara Leary, Keith Manville, Elizabeth Merkhofer, Laura Strickhart, and Matthew Walmer. Apricot: A dataset of physical adversarial attacks on object detection, 2020.

[5] Tom Brown, Dandelion Mane, Aurko Roy, Martin Abadi, and Justin Gilmer. Adversarial patch. 2017.

[6] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch, 2018.

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

[8] Ping-Yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studor, and Tom Goldstein. Certified defenses for adversarial patches, 2020.

[9] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks. *arXiv preprint arXiv:1806.11146*, 2018.

[10] Ian J Goodfellow et al. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[14] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey, 2020.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2017.

[17] Yuezun Li, Xiao Bian, Ming ching Chang, and Siwei Lyu. Exploring the vulnerability of single shot module in object detectors via imperceptible background patches, 2019.

[18] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer, 2019.

[19] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors, 2019.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[21] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[22] Camilo Pestana, Naveed Akhtar, W. Liu, D. Glance, and A. Mian. Adversarial perturbations prevail in the y-channel of the ycbcr color space. *ArXiv*, abs/2003.00883, 2020.

[23] Camilo Pestana, Wei Liu, David Glance, and Ajmal Mian. Defense-friendly images in adversarial attacks: Dataset and metrics for perturbation difficulty. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 556–565, January 2021.

[24] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[25] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[26] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L. Yuille. Adversarial attacks beyond the image space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[27] Yang Zhang, Hassan Foroosh, Philip David, and Boqing Gong. Camou: Learning physical vehicle camouflages to adversarially attack detectors in the wild. In *International Conference on Learning Representations*, 2018.