Procedural Noise Adversarial Examples for Black-Box Attacks on Deep Convolutional Networks

Kenneth T. Co
Imperial College London, UK
k.co@imperial.ac.uk

Luis Muñoz-González
Imperial College London, UK
l.munoz@imperial.ac.uk

Sixte de Maupeou
Imperial College London, UK
sd4215@imperial.ac.uk

Emil C. Lupu
Imperial College London, UK
e.c.lupu@imperial.ac.uk

ABSTRACT

Deep Convolutional Networks (DCNs) have been shown to be vulnerable to adversarial examples—perturbed inputs specifically designed to produce intentional errors in the learning algorithms at test time. Existing input-agnostic adversarial perturbations exhibit interesting visual patterns that are currently unexplained. In this paper, we introduce a structured approach for generating Universal Adversarial Perturbations (UAPs) with procedural noise functions. Our approach unveils the systemic vulnerability of popular DCN models like Inception v3 and YOLO v3, with single noise patterns able to fool a model on up to 90% of the dataset. Procedural noise allows us to generate a distribution of UAPs with high universal evasion rates using only a few parameters. Additionally, we propose Bayesian optimization to efficiently learn procedural noise parameters to construct inexpensive untargeted black-box attacks. We demonstrate that it can achieve an average of less than 10 queries per successful attack, a 100-fold improvement on existing methods. We further motivate the use of input-agnostic defences to increase the stability of models to adversarial perturbations. The universality of our attacks suggests that DCN models may be sensitive to aggregations of low-level class-agnostic features. These findings give insight on the nature of some universal adversarial perturbations and how they could be generated in other applications.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**; • **Computing methodologies → Neural networks**;

KEYWORDS

adversarial machine learning; Bayesian optimization; black-box attacks; deep neural networks; procedural noise; universal adversarial perturbations

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3345660

ACM Reference Format:
Kenneth T. Co, Luis Muñoz-González, Sixte de Maupeou, and Emil C. Lupu. 2019. Procedural Noise Adversarial Examples for Black-Box Attacks on Deep Convolutional Networks. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3319535.3345660

1 INTRODUCTION

Advances in computation and machine learning have enabled deep learning methods to become the favoured algorithms for various tasks such as computer vision [31], malware detection [62], and speech recognition [22]. Deep Convolutional Networks (DCN) achieve human-like or better performance in some of these applications. Given their increased use in safety-critical and security applications such as autonomous vehicles [5, 72, 74], intrusion detection [28, 29], malicious string detection [63], and facial recognition [39, 68], it is important to ensure that such algorithms are robust to malicious adversaries. Yet despite the prevalence of neural networks, their vulnerabilities are not yet fully understood.

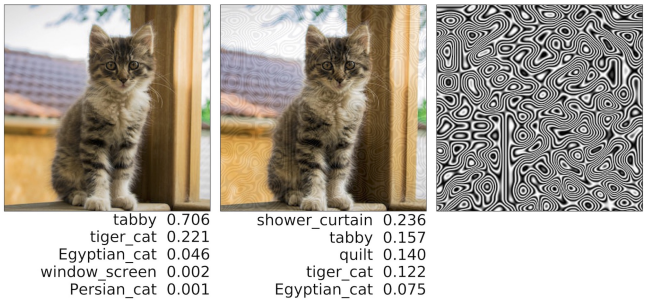

Figure 1: Adversarial example generated with a procedural noise function. From left to right: original image, adversarial example, and procedural noise (magnified for visibility). Below are the classifier's top 5 output probabilities.

It has been shown that machine learning systems are vulnerable to attacks performed at test time [2, 23, 41, 52]. In particular, DCNs have been shown to be susceptible to *adversarial examples*: inputs indistinguishable from genuine data points but designed to be misclassified by the learning algorithm [4, 71]. As the perturbation required to fool the learning algorithm is usually small, detecting adversarial examples is a challenging task. Fig. 1 shows an

adversarial example generated with the attack strategy we propose in this paper; the perturbed image of a tabby cat is misclassified as a shower curtain. Although we focus on computer vision, this phenomenon has been shown in other application domains such as speech processing [9, 11], malware classification [19], and reinforcement learning [24, 37] among others.

In this paper, we propose a novel approach for generating adversarial examples based on the use of procedural noise functions. Such functions are commonly used in computer graphics and designed to be parametrizable, fast, and lightweight [34]. Their primary purpose is to algorithmically generate textures and patterns on the fly. Procedurally generated noise patterns have interesting structures that are visually similar to those in existing universal adversarial perturbations [30, 44].

We empirically demonstrate that DCNs are fragile to procedural noise and these act as *Universal Adversarial Perturbations (UAPs)*, i.e. input-agnostic adversarial perturbations. Our experimental results on the large-scale *ImageNet* classifiers show that our proposed black-box attacks can fool classifiers on up to 98.3% of input examples. The attack also transfers to the object detection task, showing that it has an obfuscating effect on objects against the YOLO v3 object detector [58]. These results suggest that large-scale indiscriminate black-box attacks against DCN-based machine learning services are not only possible but can be realized at low computational costs. Our contributions are as follows:

- We show a novel and intuitive vulnerability of DCNs in computer vision tasks to procedural noise perturbations. These functions characterize a distribution of noise patterns with high universal evasion, and universal perturbations optimized on small datasets generalize to datasets that are 10 to 100 times larger. To our knowledge, this is the first model-agnostic black-box generation of universal adversarial perturbations.
- We propose *Bayesian optimization* [43, 66] as an effective tool to augment black-box attacks. In particular, we show that it can use our procedural noise to craft inexpensive universal and input-specific black-box attacks. It improves on the query efficiency of random parameter selection by 5-fold and consistently outperforms the popular L-BFGS optimization algorithm. Against existing query-efficient black-box attacks, we achieve a 100 times improvement on the query efficiency while maintaining a competitive success rate.
- We show evidence that our procedural noise UAPs appear to exploit low-level features in DCNs, and that this vulnerability may be exploited to create universal adversarial perturbations across applications. We also highlight the shortcomings of adversarial training and suggest input-agnostic defences to reduce model sensitivity to adversarial perturbations.

The rest of the paper is structured as follows. In Sect. 2, we define a taxonomy to evaluate evasion attacks. In Sect. 3, we describe and motivate the use of procedural noise functions. In Sect. 4, we demonstrate how different DCN architectures used in image classification have vulnerabilities to procedural noise. In Sect. 5, we show how to leverage this vulnerability to create efficient black-box attacks. In Sect. 6, we analyze how the attack transfers to the object detection task and discuss how it can generalize to other application domains. In Sect. 7, we explore denoising as a preliminary countermeasure. Finally, in Sect. 8, we summarize our findings and suggest future research directions.

2 ATTACK TAXONOMY

Our study focuses on attacks at test time, also known as *evasion attacks*. To determine the viability and impact of attacks in practical settings, we categorize them according to three factors: (a) the generalizability of their perturbations, (b) the access and knowledge the adversary requires, and (c) the desired output. These factors also describe the threat model being considered.

2.1 Generalizability

The *generalizability* of adversarial perturbations refers to their ability to apply across a dataset or to other models. Perturbations that generalize are more efficient because they do not need to be re-computed for new data points or models. Their generalizability can be described by their *transferability* and *universality*.

Input-specific adversarial perturbations are designed for a specific input against a given model, these are neither transferable or universal. *Transferable* adversarial perturbations can fool multiple models [50] when applied to the same input. This property enhances the strength of the attack, as the same adversarial input can degrade the performance of multiple models, and makes possible black-box attacks through surrogate models. Perturbations are *universal* when the same adversarial perturbation can be applied successfully across a large portion of the input dataset to fool a classifier [44]. *Cross-model universal* perturbations are both transferable and universal, i.e., they generalize across both a large portion of the inputs and across models. Generating adversarial perturbations that generalize is suitable and more efficient in attacks that target a large number of data points and models, i.e. for broad spectrum indiscriminate attacks. In contrast, input-specific attacks may be easier to craft when a few specific data points or models are targeted or for targeted attacks where the attacker aims to produce some specific types of errors.

2.2 Degree of Knowledge

For evasion attacks, we assume that the attacker has access to the test input and output. Beyond this, the adversary's knowledge and capabilities range from no access or knowledge of the targeted system to complete control of the target model. Accordingly, attacks can be broadly classified as: white-box, grey-box, or black-box [52].

In *white-box* settings, the adversary has complete knowledge of the model architecture, parameters, and training data. This is the setting adopted by many existing studies including [10, 18, 33, 40, 45, 71]. In *grey-box* settings, the adversary can build a surrogate model of similar scale and has access to training data similar to that used to train the target system. This setting is adopted in transfer attacks where white-box adversarial examples are generated on a surrogate model to attack the targeted model [32, 51]. This approach can also be adapted for a black-box setting. For example Papernot et al. [51] apply a heuristic to generate synthetic data based on queries to the target classifier, thus removing the requirement for labelled training data. In a *black-box* setting, the adversary has no knowledge of the target model and no access to surrogate datasets.

The only interaction with the target model is by querying it, this is often referred to as an "oracle".

Given the adversary's lack of knowledge, black-box attacks rely heavily on making numerous queries to gain information. This increases the chances of the attack being detected. Thus, the most dangerous attacks are those that require the least queries and resources. With fewer queries, adversarial perturbations are generated sooner, costs (when using a paid service) are lower and the volume of suspicious queries is reduced. Existing black box attacks like [3, 12] have shown some success with zeroth order optimization and gradient estimation. However they require tens to hundreds of thousands of queries on datasets with a large number of features, as in realistic natural-image dataset like ImageNet [13, 25]. The most query-efficient method reported so far is a bandit optimization framework that achieves 92.9% success with an average of $1,000$ queries per image on ImageNet [25].

2.3 Desired Output

The adversary's goal varies according to the application and the expected rewards gained from exploiting the system. Usually, attacks are considered as either *targeted* or *untargeted (indiscriminate)*.

In *targeted* attacks the adversary aims for a specific subset of inputs to be misclassified as their chosen output. In *untargeted* attacks, the attacker aims to cause classification errors on a subset of inputs. Both these attacks disrupt the machine learning system by forcing errors and undermining the model's reliability.

In multi-class classification, targeted attacks are more challenging due to their specificity, but successful ones allow a greater degree of manipulation for the attacker. On the other side, untargeted attacks are typically easier, as they just have to evade the "correct" classification, and this characteristic is more suited for broad indiscriminate attacks.

3 PROCEDURAL NOISE

We introduce *procedural noise functions* as an intuitive and computationally efficient approach to generate adversarial examples in black-box settings. Procedural noise functions are algorithmic techniques used for generating image patterns, typically used in the creation of natural details to enhance images in video and graphics production. They are designed to be fast to evaluate, scale to large dimensions, and have low memory footprint. These attributes make them desirable for generating computationally inexpensive perturbations. For a more comprehensive survey on procedural noise, we refer the reader to Lagae et al. [34].

3.1 Motivation

Existing *Universal Adversarial Perturbations (UAPs)* generated by white-box attacks exhibit interesting visual structures that are [44], as of yet, not fully understood. UAPs are particularly interesting as their universality reveals more generic or class-agnostic features that machine learning algorithms appear to be sensitive to. In contrast, input-specific adversarial perturbations, though less detectable in many cases, can "overfit" and apply only to the inputs they were designed for [77].

Current approaches, like the following, craft UAPs using white-box knowledge of the model's learned parameters. Moosavi-Dezfooli et al. [44] use the DeepFool algorithm [45] iteratively over a set of images. Mopuri et al. [46] use Generative Adversarial Nets (GANs) to compute UAPs, whilst Khrulkov and Oseledets [30] propose to use the singular vector method that maximizes the difference in activations at a targeted hidden layer between the original and the adversarial examples.

We hypothesize that procedural noise, which exhibits patterns visually similar to those of UAPs (see Appendix A), can also act as a UAP. Procedural noise is simple to implement, fast to compute, and does not require the additional overhead of building, training, or accessing a DCN to generate adversarial perturbations. The parametrization of procedural noise is simpler and this results in a reduced search space, which can enable query-efficient black-box attacks. This is particularly useful in large-scale applications like natural-image classification where existing attacks explore the entire input space, which has very high dimensionality ($\geq$ 100,000).

Procedural noise functions can be classified into three categories: lattice gradient noise, sparse convolution noise, and explicit noise [34]. Lattice gradient noise is generated by interpolating random values or gradients at the points of an integer lattice, with **Perlin noise** as a representative example. Sparse convolution noise is a sum of randomly positioned and weighted kernels,[1] with **Gabor noise** as a representative example. Explicit noise differs from the others in that the images are generated in advance and stored later for retrieval. This induces large memory costs and limits its applicability as an inexpensive attack. We therefore do not use it here and leave its investigation for future work.

3.2 Perlin Noise

We chose to use Perlin noise as a representative example for lattice gradient noise, because of its ease of use, popularity, and simplicity. Perlin noise was developed as a technique to produce natural-looking textures for computer graphics, with its initial application in motion pictures where it has remained a staple of the industry [34]. Perlin noise has a simple implementation which makes it suitable for inexpensive black-box attacks, as it is controlled by only a few parameters.

We summarize the formal construction of two-dimensional Perlin nose as described by Perlin [53, 54]. The value at a point (x, y) is derived as follows: let (i, j) define the four lattice points of the lattice square where $i = \{|x|, |x|+1\}$ and $j = \{|y|, |y|+1\}$. The four gradients are given by $q_{ij} = \mathbf{V}[\mathbf{Q}[\mathbf{Q}[i] + j]]$ where precomputed arrays $\mathbf{Q}$ and $\mathbf{V}$ contain a pseudo-random permutation and pseudo-random unit gradient vectors respectively. The four linear functions $q_{ij}(x - i, y - j)$ are then bilinearly interpolated by $s(x - |x|)$ and $s(y - |y|)$, where $s(t) = 6t^5 - 15t^4 + 10t^3$. The result is the Perlin noise value $p(x, y)$ for coordinates (x, y).

The Perlin noise function has several parameters that determine the visual appearance of the noise. In our implementation, the wavelengths λ_x, λ_y and number of octaves Ω contribute the most to the visual change. The noise value at point (x, y) with parameters $\delta_{\text{per}} = \{\lambda_x, \lambda_y, \Omega\}$ becomes

$$S_{\text{per}}(x, y) = \sum_{n=1}^{\Omega} p(x \cdot \tfrac{2^{n-1}}{\lambda_x}, y \cdot \tfrac{2^{n-1}}{\lambda_y})$$

[1] A kernel in image processing refers to a matrix used for image convolution.

To achieve more distinct visual patterns, we use a sine colour map with an additional frequency parameter ϕ_{sine}. This colour map for the noise value p is defined by $C(p) = \sin(p \cdot 2\pi\phi_{\text{sine}})$. The periodicity of the sine function creates distinct bands in the image to achieve a high frequency of edges. The resulting noise generating function G_{per} at point (x, y) is defined as the composition of our Perlin noise function and the sine colour map,

$$G_{\text{per}}(x, y) = C(S_{\text{per}}(x, y)) = \sin((S_{\text{per}}(x, y) \cdot 2\pi\phi_{\text{sine}})$$

with combined parameters $\delta_{\text{per}} = \{\lambda_x, \lambda_y, \phi_{\text{sine}}, \Omega\}$.

3.3 Gabor Noise

We use Gabor noise as a representative example for sparse convolution noise. It has more accurate spectral control than other procedural noise functions [35], where spectral control refers to the ability to control the appearance of noise as measured by its energy along each frequency band. Gabor noise can be quickly evaluated at any point in space and is characterized by only a few parameters such as orientation, frequency, and bandwidth [35]. In essence, Gabor noise is a convolution between sparse white noise and a Gabor kernel g. The Gabor kernel is the product of a circular Gaussian and a Harmonic function:

$$g(x, y) = e^{-\pi\sigma^2(x^2+y^2)} \cos[\tfrac{2\pi}{\lambda}(x\cos\omega + y\sin\omega)],$$

where σ is the width of the Gaussian, λ and ω are the period and orientation of the Harmonic function [34]. The value $S_{\text{gab}}(x, y)$ at point (x, y) is the sparse convolution with a Gabor kernel where $\{(x_i, y_i)\}$ are the random points [34].

Gabor noise is an expressive noise function and will have a large number of dimensions if each random point is assigned different kernel parameters. To simplify the implementation, we use the same parameters and weights for each random point (x_i, y_i). This results in noise patterns that have a uniform texture.

We add an additional discrete parameter ξ to control the isotropy of the Gabor noise. Having $\xi = 1$ results in anisotropic noise, which is oriented in one direction. Progressively larger ξ makes it more uniform in all directions. Implementing these changes gives an updated formulation

$$S_{\text{gab}}(x, y) = \tfrac{1}{\xi} \sum_{n=1}^{\xi} \sum_{i} g(x - x_i, y - y_i; \sigma, \lambda, \omega + \tfrac{n\pi}{\xi}).$$

To achieve high-frequency patterns and remove unwanted low-contrast oscillations, we normalize the variance spectrum of the Gabor noise using the algorithm described by Neyret and Heitz [47]. This results in min-max oscillations. Note that we refer specifically to the *frequency of edges*, i.e. how often edges appear one after the other on an image. This is different from the classical frequency used in spectral analysis. For simplicity we use "low-frequency" or "high-frequency" when referring to patterns with low or high edge frequency in the image.

We have tried applying the sine colour map to Gabor noise, but it obscures the Gabor kernel structures. Similarly, we have tried normalizing the variance spectrum of Perlin noise, but this resulted in flat images with few edges and no distinct patterns. The final noise generating function G_{per} with a normalized variance spectrum has parameters $\delta_{\text{gab}} = \{\sigma, \lambda, \omega, \xi\}$.

4 VULNERABILITY TO PROCEDURAL NOISE

In this section, we empirically demonstrate the vulnerability of five different DCN architectures to procedural noise for the ImageNet classification task. We show that randomly chosen procedural noise perturbations act as effective UAPs against these classifiers. The procedural noise UAPs greatly exceed the baseline uniform random noise, is on average universal on more than half the dataset across different models, and fools each model on more than 72% of the dataset when considering input-specific evasion.

4.1 Experiment Setup

The data used are 5,000 random images from the validation set of the ILSVRC2012 ImageNet classification task [61]. It is a widely used object recognition benchmark with images taken from various search engines and manually labelled to 1,000 distinct object categories where each image is assigned one ground truth label.

Models. We use four distinct DCN architectures pre-trained on ImageNet: VGG-19 [65], ResNet-50 [21], Inception v3 [70], and Inception ResNet v2 [69]. We abbreviate Inception ResNet v2 to **IRv2**. Inception v3 and Inception ResNet v2 take input images with dimensions $299 \times 299 \times 3$ while the remaining two networks take images with dimensions $224 \times 224 \times 3$.

We also take an ensemble adversarially trained version of the Inception ResNet v2 architecture: Tramer et al. [73] fine-tuned the IRv2 network by applying ensemble adversarial training, making it more robust to gradient-based attacks. We refer to this new model as $\textbf{IRv2}_{\textbf{ens}}$. It has the same architecture, but different weights when compared to the first IRv2. For complete details of the ensemble adversarial training process, we refer the reader to [73].

Perturbations. The Perlin noise parameters $\lambda_x, \lambda_y, \phi_{\text{sine}}$ and Gabor noise parameters σ, λ are positive and bounded above by the image's side length d. Increasing the parameters beyond this will have no impact on the resulting image as it gets clipped by the image dimension. The isotropy $\xi \in [1, 12]$ and number of octaves $\Omega \in [1, 4]$ are discrete, and evaluations show negligible change in the image for $\xi > 12$ and $\Omega > 4$. The range on the angle $\omega \in [0, 2\pi]$ covers all directions.

We fix the kernel size and number of random points for Gabor noise so that the Gabor kernels always populate the entire image. The parameters δ_{gab} of the Gabor noise have the greater influence on the resulting visual appearance of the noise pattern.

To provide a baseline, we also test the models against uniform random noise perturbations: $\text{sgn}(r) \cdot \varepsilon$ where $r \in \mathcal{U}(-1, 1)^{d\times d\times 3}$, d is the image's side length, and ε is the ℓ_∞-norm constraint on the perturbation. This is an ℓ_∞-optimized uniform random noise, and it is reasonable to say that attacks that significantly outperform this baseline are non-trivial.

Metrics. To measure the universality of a perturbation, we define the *universal evasion rate* of a perturbation over the dataset. Given model output f, input $x \in X$, perturbation s, and small $\varepsilon > 0$, the universal evasion of s over X is

$$\frac{|\{x \in X : \arg\max f(x + s) \neq \tau(x)\}|}{|X|}, \quad \|s\|_\infty \leq \varepsilon,$$

where $\tau(x)$ is the true class label of x. An ℓ_∞-norm constraint on s ensures that the perturbation is small and does not drastically alter the visual appearance of the resulting image; this is a *proxy*

for the constraint $\tau(x) = \tau(x+s)$. We choose the ℓ_∞-norm as it is straightforward to impose for procedural noise perturbations and is often used in the adversarial machine learning literature. In this case, f is the probability output vector, but note that the attacker only needs to know the output class label $\arg\max f(x+s)$ to determine if their attacks succeed. To measure the model's sensitivity against the perturbations for each input, we define the model's *average sensitivity* on an input x over perturbations $s \in S$ as

$$\frac{|\{s \in S : \arg\max f(x+s) \neq \tau(x)\}|}{|S|}, \quad \|s\|_\infty \leq \varepsilon.$$

This will help determine the portion of the dataset on which the model is more vulnerable. Finally, the *input-specific evasion* rate is

$$\frac{|\{x \in X : \exists s \in S \text{ such that } \arg\max f(x+s) \neq \tau(x)\}|}{|X|}, \quad \|s\|_\infty \leq \varepsilon.$$

This measures how many inputs in the dataset can be evaded with perturbations from S.

Experiment. We evaluate our procedural noise perturbations on 5,000 random images from the validation set. The perturbations generated are from 1,000 Gabor noise, 1,000 Perlin noise, and 10,000 uniform random perturbations. 1,000 queries for the procedural noise functions was sufficient for our results and its search space only has four bounded parameters. We use an ℓ_∞-norm constraint of $\varepsilon = 16$. The pixels of the perturbations are first clipped to $[-\varepsilon, \varepsilon]$ and the resulting adversarial example's pixels are clipped to the image space $[0, 255]$.

Thus, this is an untargeted black-box attack with exactly 1,000 queries per image for procedural noise. Note that the adversary has no knowledge of the target model and only requires the top label from the model's output.

4.2 Universality of Perturbations

The procedural noise perturbations appear to be both universal and transferable across models. The results in Fig. 2 show that the models in order from least to most robust against the perturbations are: VGG-19, ResNet-50, Inception v3, IRv2, and then IRv2_{ens}. This is not surprising, as the generalization error for these models appear in the same order, with larger generalization error indicating a less robust model. The ensemble adversarially trained model has also been hardened against adversarial examples, so it was expected to be less affected. However, the ensemble adversarial training did not fully mitigate the impact of the procedural noise.

Both Gabor and Perlin noise have significantly higher universal evasion rate than random noise. In Fig. 2a, we represent random noise using the median (which, in this case, is very close to the mean) of its universal evasion rate over all 10,000 perturbations, as the variance is very small (less than 10^{-5}) for each model.

Procedural noise functions create a distribution whose modes have high universal evasion rates. Table 1 shows an example of this on Inception v3, where more than half the Perlin noise perturbations achieve evasion on more than 57% of the dataset.

Between the two procedural noises, Perlin noise is a stronger UAP, though not by a large margin. In Fig. 2a, Gabor noise appears bimodal, especially with the IRv2 classifiers. This bimodal aspect shows that there are two large disjoint subsets of Gabor noise: one with high universal evasion and another with low universal evasion. This was not as prominent for Perlin noise.

Table 1: Universal evasion (%) by percentile for random and procedural noise perturbations on Inception v3.

Percentile	Random	Gabor	Perlin
min	26.0	24.7	**26.4**
25th	26.8	39.9	**47.2**
50th	27.0	49.1	**57.7**
75th	27.1	53.3	**63.7**
max	27.9	61.8	**73.2**

Best Parameters. Given the distribution of universal evasion rates in Fig. 2a, it is interesting to observe which parameters of the noise functions contribute most to the evasion rate. Because there are only four parameters for each procedural noise function, the linear correlations between these parameters and the universal evasion rates give a general overview of their influence. Though this analysis may not fully capture multi-variable relationships.

For Gabor noise, the parameters σ, ω, and ξ have low correlation (≤ 0.16) with universal evasion. The remaining parameter λ however has a high correlation (≥ 0.52) with the universal evasion rate across all classifiers; this peaks at 0.90 for IRv2_{ens}. λ corresponds to the wavelength of the harmonic kernel. Visually, lower values of λ indicate thinner and more frequent bands, which is why λ can also be thought of as an inverse frequency. The correlations suggest that low-frequency Gabor noise patterns correlate with high universal evasion, and this appears to be more pronounced as we move to the more robust models.

For Perlin noise, the parameters λ_x and λ_y have low correlation (≤ 0.25) with universal evasion. The universal evasion rates have a moderate negative correlation with the number of octaves Ω and have a moderately-high positive correlation with ϕ_{sine} for the non-adversarially trained models. This suggests that high-frequency Perlin noise patterns correlate with high universal evasion. The number of octaves indicate the amount of detail or the number of curvatures within the image, so a slight negative correlation indicates that some patterns that achieve high evasion have fewer curvatures and details. Visually, the frequency of the sine function ϕ_{sine} can be thought of as the thinness of the bands in the image. This is the opposite of the λ in Gabor noise, so we have a similar result where IRv2_{ens} is more susceptible to low-frequency patterns.

Low-frequency patterns are also more effective on IRv2_{ens} because this model was adversarially trained on gradient-based attacks, which we hypothesize generated mostly *high-frequency* perturbations. The frequency appears to be an important factor in determining the strength of the UAP even when Gabor and Perlin noise have opposite correlations between their frequency and their universal evasion. This difference shows that these noise patterns have notably different characteristics. Complete correlation matrices are available in Appendix D.

Cross-model Universality. The UAPs with high universal evasion on one model often have high universal evasion on the other models. For example, a Perlin noise perturbation has 86.7%, 77.4%, 73.2%, 58.2%, and 45.9% universal evasion on VGG-19, ResNet-50, Inception v3, IRv2, and IRv2_{ens} respectively.

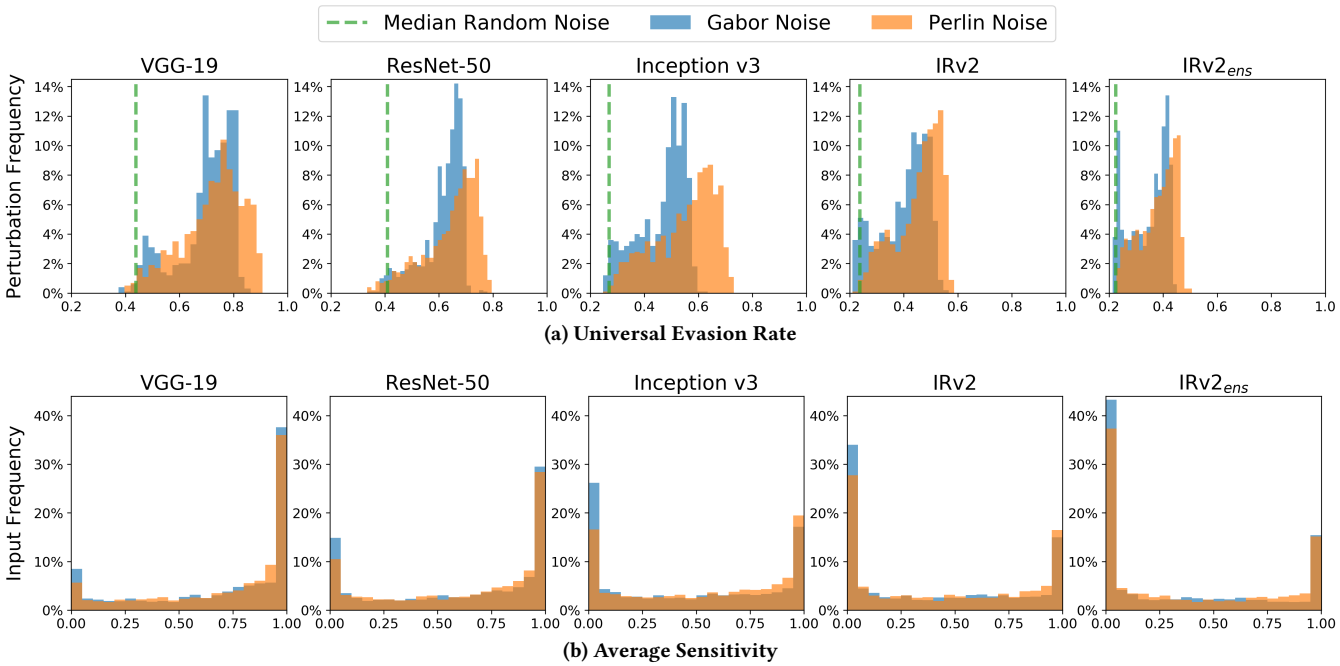

Figure 2: Histogram of (a) universal evasion rate over all perturbations and (b) average sensitivity over all inputs across all models. For example, VGG-19 in (a) shows that about 12% of the Gabor noise perturbations have a universal evasion rate of approximately 0.8.

The correlation of the universal evasion rates across models is high (≥ 0.80), which shows the transferability of our UAPs across models. For Gabor noise, there is a high correlation across all models. For Perlin noise, there appears to be a noticeable difference between IRv2_{ens} and the remaining four models. The strongest Perlin noise on the undefended models have high-frequency patterns that the adversarial training partially mitigated.

Overall, the results show that procedural noise perturbations act as UAPs across all models. This is the first black-box generation of cross-model UAPs, and the procedural noise allows us to draw perturbations from a distribution with high average universal evasion. We now look at the results from an input-specific perspective.

4.3 Model Sensitivity on Inputs

The model's sensitivity could vary across the input dataset, i.e. the model's predictions are stable for some inputs while others are more susceptible to small perturbations. We measure this with the average sensitivity of single inputs over all perturbations. Results in Fig. 2b show that the average sensitivity of the dataset is bimodal for all models and both procedural noise functions. There are two distinct subsets of the data: one on the right that is very sensitive and the other on the left that is very insensitive to the perturbations. The remaining data points are somewhat uniformly spread in the middle. The number of points on the left peak of the histogram in Fig. 2b is larger for the most robust models. Similarly to Fig. 2a, this progression indicates that the order of most to least sensitive models align with the most to least robust models. We omit results for random noise since more than 60% of the input dataset are barely affected by it across all models. This manifests as a tall peak on the left and a short peak on the right.

When comparing the average sensitivity of inputs between the two procedural noise functions on the same models, the correlations range between 0.89-0.92, which shows that both procedural noise perturbations affect very similar groups of inputs for each model. The correlation between the average sensitivities for each input across the Inception models is at least 0.79, which suggests that these models are sensitive to procedural noise on similar inputs. This is less so between ResNet-50 and VGG-19 whose correlations with the other models range from 0.56-0.81.

Input-specific Evasion. We consider the case when our untargeted black-box attack is used as an input-specific attack, i.e. the adversary only needs to find at least one adversarial perturbation that evades each input. Thus, Evasion on input x is achieved when $\exists s \in S$ such that $\arg\max f(x+s) \neq \tau(x)$. This is in contrast to the universal attack where the adversary crafts a single perturbation to fool the model on as many inputs as possible.

Note that the random noise is optimized for the ℓ_∞-norm. We draw the random noise perturbation from $\{-\varepsilon, \varepsilon\}^{d\times d\times 3}$. Thus, for each pixel the added noise is either $-\varepsilon$ or ε, rather than drawing from the continuous domain $(-\varepsilon, \varepsilon)$. It is reasonable to think that a larger perturbation is more likely to cause evasion.

Table 2 shows that VGG-19 and ResNet-50 are particularly fragile as even random noise greatly degrades their performance. Although the Inception models are more robust, both procedural noise perturbations still achieve more than 72% evasion on all of them. Although ensemble adversarial training improved the robustness of IRv2, it

still does not mitigate the impact of the procedural noise attack. The idea of ensemble adversarial training was to decouple the generation of the adversarial training set from the original model, but this was limited since it was only done for gradient-based attacks. We argue that defences should be more input-agnostic to avoid having to train against all types of attacks.

Table 2: Input-specific evasion rate (in %) for random and procedural noise perturbations. Original refers to the top 1 error on the unaltered original images. Strongest attack on each classifier is highlighted.

Classifier	Original	Random	Gabor	Perlin
VGG-19	29.4	57.1	97.7	**98.3**
ResNet-50	25.9	55.7	96.2	**96.3**
Inception v3	22.3	46.8	89.2	**93.6**
IRv2	20.1	38.7	81.1	**87.0**
$IRv2_{ens}$	20.1	37.5	72.7	**79.4**

Label Analysis. The procedural noise perturbations were not designed to be a targeted attack, but intuitively, the same universal perturbation would cause misclassification towards class labels that have visually similar textures to the procedural noise pattern. We find that this holds true for only a few of the procedural noise UAPs.

For a given procedural noise perturbation, we define its *top label* to be the class label it causes the most misclassification towards. When looking at procedural noise UAPs across all models, about 90% of Gabor noise perturbations have their top label apply to at most 9% of the inputs. For Perlin noise, about 80% of its perturbations have their top label on at most 10% of the input. There are however a few outliers that have their top label appear above 10% of the inputs. For example, on Inception v3, the top Gabor noise with 61.8% universal evasion has "window screen" as its top label and it applies for 37.8% of the evaded inputs. In contrast, another Gabor noise perturbation with 58.9% universal evasion has "quilt" as its top label, but it only applies for 6.0% of the evaded inputs. As a consequence, it is still possible to use procedural noise to create universal targeted UAPs aimed at specific class labels like "window screen" or "brain coral". However the class labels we can target is dependent on the procedural noise and the overall success of universal targeted attacks may be limited, as it is more difficult to make a perturbation both universal and targeted.

Perlin noise has a relatively large amount of "brain coral" classifications, with other labels such as "maze" and "shower curtain" also appearing frequently in the top five most classified labels per classifier. For Gabor noise, there was no label that consistently appeared at the top across all models. These results indicate that Perlin noise has a larger bias towards certain classes, while Gabor noise is more indiscriminate.

4.4 Discussion

Adversarial examples exploit the model's sensitivity, causing large changes in the model's output by applying specific small changes to the input. More specifically, recent work has shown adversarial examples exploit "non-robust" features that the model learns but that are incomprehensible to humans [26]. It is likely that UAPs may be exploiting "universal" non-robust features that the DCN has learned.

Textures. Previous results have shown ImageNet-trained DCNs to be more reliant on textures rather than shapes [14]. Though not explicitly shown in the later Inception architectures (Inception v3, IRv2), these are still likely to have a strong texture-bias due to similarities in the training. The texture bias however does not fully explain why small, and sometimes imperceptible, changes to the texture can drastically alter the classification output. We attribute adversarial perturbations more to the sensitivity of the model. Moreover, we test our attack against an object detection model in Sect. 6, and show that it also degrades the performance of models that have a spatial component in their learning task.

Generalizability of Procedural Noise. From the label analysis of procedural noise UAPs, we observe that most perturbations with high universal evasion do not have a strong bias towards any particular class label—no particular class is targeted more than 10% for over 80% of the procedural noise UAPs. This suggest that UAPs leverage more generic low-level features that the model learns, which would explain their *universality* and indiscriminate behaviour in causing misclassification.

Amongst white-box UAP attacks, our procedural noise has the closest visual appearance to perturbations from the Singular Vector Attack (SVA) by Khrulkov and Oseledets [30]. They generate UAPs targeting specific layers of DCNs, and found that targeting earlier layers of the network generated more successful UAPs. The patterns obtained for these earlier layers share a visual appearance with procedural noise. These layers also correspond to the low-level features learned by the network. Other evidence also suggests that procedural noise exploits low-level features, as feature visualization of earlier layers in neural networks share the same visual appearance with some procedural noise patterns. Convolutional layers induce a prior on DCNs to learn local spatial information [17], and DCNs trained on natural-image datasets learn convolution filters that are similar in appearance to Gabor kernels and colour blobs [48, 76]. Gabor noise appears to be a simple collection of low-level features whereas Perlin noise seems to be a more complex mixture of low-level features. This difference in the complexity of their visual appearance may explain why Perlin noise is a stronger attack than Gabor noise.

Procedural noise attacks transfer with high correlation across the models most likely because they share the same training set (ImageNet), learning algorithms (e.g. backpropagation), and have similar components in their architectures (e.g. convolutional layers). This increases the likelihood that they share input-agnostic vulnerabilities. In this way, our results appear to support the idea that DCNs are *sensitive* to aggregations of low-level features.

Security Implications. In transfer learning, a model trained on one task is re-purposed as an initialization or fixed feature extractor for another similar or related task. When used as a feature extractor, the initial layers are often frozen to preserve the low-level features learned. The subsequent layers, closer to the output, are then re-trained for the new task [76]. Hidden layers from models pre-trained on the ImageNet dataset are often re-used for other natural-image classification tasks [49]. This makes it a notable target as vulnerabilities that exploit low-level features encoded in the earlier layers carry over to the new models.

Transfer learning has many benefits as training entire models from scratch for large-scale tasks like natural-image classification can be costly both computationally in training and in terms of gathering the required data. However, this creates a systemic threat, as subsequent models will also be vulnerable to attacks on low-level features like procedural noise. Precisely characterizing the extent of the vulnerability of re-purposed models to the same attack is an interesting direction for future work.

Procedural noise is an accessible and inexpensive way for generating UAPs against existing image classifiers. In our experiments the Gabor and Perlin noise functions modified only four parameters, and each parameter was bounded in a closed interval. Drawing from this space of perturbations has generated UAPs with high universal evasion rates. Attackers can take advantage of the small search space by using procedural noise to craft query-efficient untargeted black-box attacks as we will show in Sect. 5.

Strengths & Limitations. Procedural noise is one of the first black-box generation of UAPs. Other existing black-box attacks often optimize for input-specific evasion, and other existing UAP generation methods are white-box or grey-box. Whilst procedural noise is also a generative model, it differs from other generative models like Bayesian networks, Generative Adversarial Networks (GANs), or Variational Autoencoders (VAEs) as it does not require the additional overhead of building and training these generative models–which often requires more resources and stronger adversaries to execute successfully.

Comparing procedural noise attacks with existing black-box attacks is not straightforward, as procedural noise naturally has high universal evasion rates. This gives procedural noise an advantage in that, despite having no access to the target model, randomly drawn procedural noise patterns are likely to have high universal evasion. The search space for our procedural noise has only four dimensions, whereas most attacks are designed for the whole input space of hundreds of thousands of dimensions. However, this does come at a cost, as procedural noise functions do not capture adversarial perturbations outside their codomain. Other attacks that explore the whole input space are able to take full advantage of a stronger adversary (i.e. white-box or grey-box setting) or larger query limits. We explore this trade-off further with input-specific black-box attacks in Sect. 5.3.

Another limitation of this experiment was the use of an ℓ_∞-norm constraint. Although it is often used in the literature as a proxy for human perception, ℓ_p-norms have limitations [16] e.g., low-frequency patterns appear to be more visible than high-frequency patterns for the same ℓ_p-norm. This frequency could be used as an additional constraint and developing a more reliable proxy for human perception remains an interesting avenue for future work.

Summary. Procedural noise attacks specialize as untargeted black-box perturbations with naturally high universal evasion. Offensively, this has applications as a large-scale indiscriminate attack and, defensively, as a standard test on machine learning services. We expand on this perspective in the following sections. In Sect. 5, we develop query-efficient black-box attacks using procedural noise and Bayesian optimization. In Sect. 6, we apply our procedural noise attack against a DCN designed for object detection, showing that the attack can generalize to other tasks. In Sect. 7, we test an input-agnostic defence in median filter denoising.

5 EFFICIENT BLACK-BOX ATTACKS

Whilst in previous sections we have shown that procedural noise functions are an efficient way to generate adversarial perturbations, another significant advantage they bring is their low-dimensional search space. This enables the use of query-efficient black-box optimization techniques that otherwise do not scale well to high-dimensional problems. In this section, we compare several black-box optimization techniques for both input-specific and universal attacks and show that Bayesian optimization is an efficient method for choosing parameters in such black-box attacks.

5.1 Bayesian Optimization

Bayesian optimization is a sequential optimization algorithm used to find optimal parameters for black-box objective functions [43, 66]. This technique is often effective in solving various problems with expensive cost functions such as hyperparameter tuning, reinforcement learning, and combinatorial optimization [64]. Bayesian optimization consists of a probabilistic surrogate model, usually a Gaussian Process (GP), and an acquisition function that guides its queries. GP regression is used to update the belief on the parameters with respect to the objective function after each query [64].

Gaussian Processes. A GP is the generalization of Gaussian distributions to a distribution over functions and is typically used as the surrogate model for Bayesian optimization [55]. We use GPs as they induce a posterior distribution over the objective function that is analytically tractable. This allows us to update our beliefs about the objective function after each iteration [66].

A Gaussian Process $\mathcal{GP}(m, k)$ is fully described by a prior mean function $m : \mathcal{X} \to \mathbb{R}$ and positive-definite kernel or covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We describe GP regression to understand how our GP is updated when more observations are available. The following expressions give the GP prior and Gaussian likelihood respectively

$$p(f \mid \mathbf{X}) = \mathcal{N}(m(\mathbf{X}), \mathbf{K})$$
$$p(y \mid f, \mathbf{X}) = \mathcal{N}(f(\mathbf{X}), \sigma^2\mathbf{I})$$

where $\mathcal{N}$ denotes a normal distribution. Elements of the mean and covariance matrix are given by $m_i = m(\mathbf{x}_i)$ and $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Given observations $\{\mathbf{X}, \mathbf{y}\}$ and an arbitrary point $\mathbf{x}$, the updated posterior mean and covariance on the n-th query are given by

$$m_n(\mathbf{x}) = m(\mathbf{x}) - (\mathbf{K} + \sigma^2\mathbf{I})^{-1}(y - m(\mathbf{X}))$$
$$k_n(\mathbf{x}, \mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{X})(\mathbf{K} + \sigma^2\mathbf{I})^{-1}k(\mathbf{X}, \mathbf{x}).$$

We take the mean function to be zero $m \equiv 0$ to simplify evaluation and since no prior knowledge can be incorporated into the mean function [64] as is the case for black-box settings.

A GP's ability to model a rich distribution of functions rests on its covariance function which controls important properties of the function distribution such as differentiability, periodicity, and amplitude [55, 64]. Any prior knowledge of the target function is encoded in the hyperparameters of the covariance function. In a black box setting, we adopt a more general covariance function in the Matérn 5/2 kernel

$$k_{5/2}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right)$$

where $r = \mathbf{x} - \mathbf{x}'$ and l is the length-scale parameter [66]. This results in twice-differentiable functions, an assumption that corresponds to those made in popular black-box optimization algorithms like quasi-Newton methods [66].

Acquisition Functions. The second component in Bayesian optimization is an acquisition function that describes how optimal a query is. Intuitively, the acquisition function evaluates the utility of candidate points for the next evaluation [7]. The two most popular choices are the *Expected Improvement* (EI) and *Upper Confidence Bound* (UCB) [64]. First we define $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ as the predictive mean and variance of $g(\mathbf{x})$ respectively. Let $\gamma(\mathbf{x}) = \frac{g(\mathbf{x}_{\text{best}}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$. The acquisition functions are

$$\alpha_{\text{EI}}(\mathbf{x}) = \sigma(\mathbf{x})(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}) \mid 0, 1))$$
$$\alpha_{\text{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \quad \kappa > 0$$

where Φ is the normal cumulative distribution function.

EI and UCB have both been shown to be effective and data-efficient in real black-box optimization problems [66]. However, most studies have found that EI converges near-optimally and is better-behaved than UCB in the general case [7, 64, 66]. This makes EI the best candidate for our acquisition function.

5.2 Universal Black-box Attack

For a universal black-box attack using procedural noise, the goal is to find the optimal parameters δ^* for the procedural noise generating function G so that the universal evasion rate of perturbation $G(\delta^*)$ generalizes to unknown inputs X_{val}. The attacker has a smaller set of inputs X_{train} and optimizes their perturbation δ^* for this dataset. The performance of the attack is measured by its universal evasion rate over the validation set X_{val}. In a practical setting, this is where the attacker optimizes their procedural noise UAP over a small dataset, then injects that optimized perturbation to other inputs–with the goal of causing as many misclassifications as possible.

Experiment. We use the Inception v3 model, ℓ_∞-norm $\varepsilon = 16$, and the same 5,000 data points tested in Sect. 4 as X_{val}. X_{train} are points from the ILSVRC2012 validation set not in X_{val}. We test for training set sizes of 50, 125, 250, and 500, which corresponds to 1%, 2.5%, 5%, and 10% of the validation set size.

We compare Bayesian optimization with Limited-memory BFGS (L-BFGS) [38], a quasi-Newton optimization algorithm that is often used in black-box optimization and machine learning. As the procedural noise functions are non-differentiable, we estimate gradients using finite difference approximation. This gradient approximation is similar to what is used for other black-box attacks like zeroth-order optimization [12], but here it is applied to a significantly smaller search space. When L-BFGS converges, possibly to a local optima, we restart the optimization with a different random initial point, stopping when the query limit is reached and choosing the best optima value found.

We set a maximum query limit of 1,000 universal evasion evaluations on the training set X_{train}. In practice, this limit was not necessary as both algorithms converged faster to their optimal values: within the first 100 queries for Bayesian optimization and within the first 250 queries for L-BFGS. These are untargeted universal black-box attacks where the adversary has no knowledge of the target model and only requires the top label from the model's outputs.

Results. The best procedural noise UAP computed from the training sets generalized well to the much larger validation set, consistently reaching 70% or more universal evasion on the validation set for Perlin noise. This is a surprising result as the training sets were 10-100 times smaller than the validation set. This may be due to the inherent universality of our procedural noise perturbations. We focus on Perlin noise as it outperforms Gabor noise, with the latter averaging 58% universal evasion on the validation set.

Table 3 shows that Bayesian optimization (BayesOpt) reliably outperforms L-BFGS in terms of universal evasion rate on the training sets and the resulting universal evasion rates on the validation set. For comparison, random parameters for Perlin noise in Sect. 4 had a 98th percentile of 70.2% and a maximum of 73.1% universal evasion. Bayesian optimization consistently reached or passed this 98th percentile while L-BFGS did not. It is reasonable for these optimization algorithms not to beat the maximum since the training sets were significantly smaller. Similar trends appear between Bayesian optimization, random selection, and L-BFGS for Gabor noise perturbations. We include these results in the Appendix C.

Table 3: Comparison on Inception v3 between universal Perlin noise black-box attacks. Universal evasion rates (%) of the optimized perturbations are shown for their respective training set and the validation set.

	$\text{BayesOpt}_{\text{per}}$		$\text{L-BFGS}_{\text{per}}$	
Train Size	Train	Val.	Train	Val.
50	78.0	71.4	74.0	69.9
125	77.6	70.2	76.0	71.5
250	71.6	71.2	71.2	69.7
500	75.0	**72.9**	73.4	70.8

5.3 Input-specific Black-box Attack

In this section, we use procedural noise for an input-specific black-box algorithm. The goal of the adversary is to evade as many inputs in the dataset, maximizing the input-specific evasion rate on inputs that are not misclassified by the model. An important metric here is the query-efficiency of these attacks, as requiring large volumes of queries *per sample* becomes impractical in real-world scenarios.

Metrics. We define the *success rate* of an attack to be its input-specific evasion excluding clean inputs that are already misclassified. The *average queries* is measured over successful evasions.

Experiment. We use the Inception v3 model, ℓ_∞-norm $\varepsilon = 16$, and the same 5,000 data points from Sect. 4. For a given input x, the goal is to achieve evasion with $\arg\max f(x+s) \neq \tau(x)$ by minimizing the probability of the true class label $\tau(x)$. In this case we allow the attacker to access the model's output probability vector, as the black-box optimization algorithms gain minimal usable information if the output is binary ($\tau(x)$ or $\neg\tau(x)$).

As in the previous section, we compare Bayesian optimization and L-BFGS with a limit of 1,000 queries per input and number of restarts when it converges. We use sparse GPs to better scale Bayesian optimization [42], as the standard GP scales cubically

Table 4: Comparison on Inception v3 between our input-specific Perlin noise black-box attacks and bandit attack [25] for different query limits.

Attack	Query Limit	Average Queries	Success Rate (%)
$\text{BayesOpt}_{\text{per}}$	100	7.0	91.6
$\text{BayesOpt}_{\text{per}}$	1,000	**8.4**	**92.8**
$\text{L-BFGS}_{\text{per}}$	100	19.8	71.7
$\text{L-BFGS}_{\text{per}}$	1,000	70.1	86.5
$\text{Random}_{\text{per}}$	1,000	36.3[a]	91.6
$\text{Bandits}_{\text{TD}}$	100	29.0	36.7
$\text{Bandits}_{\text{TD}}$	1,000	224	73.6
$\text{Bandits}_{\text{TD}}$	10,000	**888**	**96.9**

[a]For each input, we divide the total number of perturbations by the number of those that evade that input to get an expected number of queries.

with the number of observations. We also compare with uniform random parameter selection from Sect. 4. These are untargeted input-specific black-box attacks where the adversary has no knowledge of the target model and only requires the output probability vector of the model.

Results. Table 4 shows that Bayesian optimization reached a high 91.6% success rate with with just 7 queries per successful evasion on average, under a restrictive 100 query limit. This vastly improves on the query efficiency over random parameters by 5-fold whilst attaining the same accuracy. The improvement for Bayesian optimization increasing the query limit from 100 to 1,000 was very incremental. This suggests that the limited codomain of the procedural noise function is setting an upper bound on the attack's success rate.

L-BFGS performed the worst as we observe it can get trapped and waste queries at *poor* local optima, which requires several restarts from different initial points to improve its performance. There were similar trends for Gabor noise where Bayesian optimization had 78.3% success with 9.8 queries on average. For both procedural noise functions, Bayesian optimization improved the average query efficiency over L-BFGS and random parameter selection by up to 7 times while retaining a success rate greater than 83%. We include the other results in Appendix C.

Comparison. We compare our results with Ilyas et al. [25], where they formalize their attack as a gradient estimation problem (like in white-box gradient-based attacks), and use a bandit optimization framework to solve it. We test the bandits attack on the same model, dataset, and ℓ_∞-norm for maximum query limits of 100, 1,000, and 10,000.

The results show that the evasion rate of our input-specific procedural noise attack greatly outperforms the bandits attack when the query limit per image is a thousand or less. For significantly larger query limits, the Perlin noise Bayesian optimization attack has a competitive success rate at a drastically better query efficiency–needing *100* times less queries on average.

The procedural noise attack is optimized for universal evasion and a restrictively small amount of queries. Most existing methods explore the entire image space which has a hundred thousand dimensions, and some attacks reduce their search space with techniques like tiling, but the dimensionality is still much larger than the four parameters of our procedural noise. Other input-specific black-box attacks [3, 6, 12] require tens to hundreds of thousands of queries on realistic natural-image datasets, which makes them inefficient, but almost certain to find an adversarial example given enough queries. The bandits method makes a small sacrifice in success rate for more query efficiency, and our procedural noise attack takes it further for greater query-efficiency.

Procedural noise perturbations have naturally high evasion rates, although the expressiveness of our chosen functions can be less than attacks whose larger codomains can capture more kinds of adversarial perturbations. These other attacks make this trade-off by sacrificing efficiency in black-box scenarios, as the number of queries they need to craft successful adversarial examples is large. On the other hand, we can increase the expressiveness of our procedural noise functions by introducing more parameters by, for example, using different colour maps. However, this may come at the cost of its naturally high universal evasion.

Summary. Black-box optimization techniques like Bayesian optimization are able to take advantage of the drastically reduced search space of procedural noise. Together with its naturally high universal evasion, we show that procedural noise gives rise to inexpensive yet potent untargeted black-box attacks, whether it be input-specific or universal. It is also shown to be competitive with existing attacks that often require orders of magnitude more queries or resources.

6 OTHER APPLICATIONS

In this section, we show how procedural noise attacks extend to object detection DCNs by attacking *YOLO v3* model. We then discuss how exploiting the sensitivity of DCNs to low-level features, like in procedural noise attacks, can be generalized to craft universal attacks for DCNs in other application domains.

6.1 Attacking Object Detection

Object detection requires the identification of locations and classes of objects within an image. "Single Shot Detectors" (SSD) such as the different versions of YOLO [56, 57] generate their predictions after a single pass over the input image. This allows SSDs to process images in real-time speeds while maintaining high accuracy. Other types of object detectors such as Fast R-CNN [15] and Faster R-CNN [59] are based on region proposals. They have comparable accuracy but do not achieve the same image processing speed as SSDs. We test our attack against YOLO v3 [58], the latest iteration of YOLO, which achieves high accuracy and detects objects in real time.

Metrics. We use precision, recall, and mean average precision (mAP) as our primary metrics; mAP is frequently used when comparing the overall performance of object detectors. In object detection the classifier predicts a bounding box to identify the location of an object and assigns that box a class label. The *Intersection Over Union (IOU)* is the ratio of the intersection area over the union area between the predicted and ground truth bounding boxes. A threshold is usually set to determine what IOU constitutes a positive classification. True positives occur when the IOU is larger than the threshold and the class is identified correctly. False negatives occur when the threshold is not met with the correct class for a ground truth bounding box. False positives occur when the IOU

is less than the threshold, there are no intersecting ground truth boxes, or there are duplicate predicted bounding boxes.

Experiment. We use the MS COCO dataset [36] which contains 80 different classes, with a large proportion of targets being the "person" class. This has become one of the benchmark datasets for object detection. We use standard settings as described in [58], with input dimensions $416 \times 416 \times 3$ and an IOU threshold of 0.5.

We use ℓ_∞-norm $\varepsilon = 16$ and apply each of our procedural noise perturbations on the 1,000 random images from the validation set. The perturbations generated are from 1,000 Gabor noise, 1,000 Perlin noise, and 1,000 ℓ_∞-optimized uniform random perturbations. The parameters are chosen uniformly at random so that we can analyze the results, as in Sect. 4. This is a universal untargeted black-box attack.

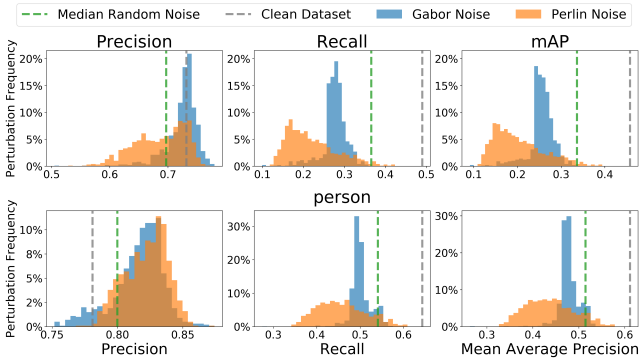

Figure 3: Histogram of metrics over all perturbations on YOLO v3. Results for (top) all classes and (bottom) the "person" class.

Results. The procedural noise perturbations significantly decrease the mAP of the model, with 75% of Perlin noise perturbations halving the mAP or worse. The precision and recall also decreased in the presence of our perturbations. Again, Perlin noise outperforms Gabor noise on average but Gabor noise has a smaller minimum, with at least one perturbation causing 0.09 mAP. Compared to uniform random noise, which maintained around 0.34 mAP, both Perlin and Gabor noise had larger impact on the model's performance. We represent uniform random noise using the median (which, in this case, is very close to the mean) on each metric as the variance is very small (less than 10^{-5}).

In Fig. 3, for the metrics across all classes, we can observe that the precision was maintained or decreased while the recall and mAP decreased. Some perturbations decreased all three metrics, which indicate that these cause both false positives and false negatives consistently. When the perturbation increases the precision while decreasing the recall and mAP, then the noise is likely masking objects rather than introducing new objects to the image. This masking effect has serious implications for security applications like surveillance and autonomous vehicles.

We focus on the "person" class as it constitutes a majority of the targets and is semantically meaningful in the context of some applications. In Fig. 3, metrics for "person" follow a similar trend to the metrics across all classes. However, the increase in precision is very small (<0.10) compared to the large drops in recall and mAP caused by Perlin noise. The higher precision indicates fewer false positives, while the decrease in recall indicates more false negatives. This indicates that the classifier is making fewer predictions overall, which means that the noise is masking persons in the image.

Whilst for the most relevant "person" class, our procedural noise appears to have an obfuscating effect, for other classes like "zebra" all the metrics decrease – indicating that there are many false positive and false negatives. However, for three classes, "backpack", "book", and "toaster", all three metrics improve. These labels did not have as much representation in the test set which may explain this anomalous result.

The frequency of the sine ϕ_{sine} for Perlin noise had the largest inverse correlation of less than -0.72 with each of the three metrics. This means that high-frequency patterns decrease the model's performance metrics, similarly to what we have observed for the image classifiers. The thickness λ of Gabor noise is moderately correlated at 0.4 with the precision, but not the recall or mAP. This suggests that thicker Gabor noise perturbations decrease the number of false positives relative to the other perturbations. We include complete correlation matrices in Appendix D.

Discussion. Object detection is a more complex task than image classification as it has to identify multiple objects and their locations within an image. Although YOLO v3 has a different architecture, task, and dataset from the ImageNet classifiers, we see that the same procedural noise perturbations are still able to greatly degrade its performance. This shows that it is more likely caused by the models' sensitivity towards perturbations rather than a texture bias, as the object detection task has a spatial component to it. This suggests that our procedural noise attacks may generalize to other DCNs on computer vision tasks with natural-image data.

In our discussion in Sect. 4.4, we hypothesized that the procedural noise perturbations are an aggregation of low-level features at high frequencies that the DCNs strongly respond to. The prior that convolutional layers induce and similarities across natural-image datasets may be why DCNs are sensitive to these noise patterns. Additionally, like in Sect. 5, we can also create more efficient black-box attacks by applying black-box optimization techniques such as Bayesian optimization to enhance the attack against object detectors. When using Bayesian optimization, an attacker can focus on minimizing a specific metric (precision, recall, mAP, or F1 score).

6.2 Beyond Images

One of the main strengths of procedural noise is that it allows to describe a distribution of UAPs with only a few parameters. This compressed representation allows for an efficient generation of UAPs both for undermining a machine learning service or, defensively, for testing the robustness of a model. In practice, compressed representations can be learned by generative models such as GANs [46, 75] or Variational Autoencoders (VAEs), however these incur additional training, calibration, and maintenance costs. Training algorithms or defences that incorporate domain-specific knowledge may be needed to mitigate the sensitivity towards attacks that make use of these compact representations.

DCNs in other applications may also be vulnerable to aggregations of low-level features due to the use of convolutional layers.

Future attacks can exploit how DCNs rely on combining low-level features rather than understanding the more difficult global features in the input data. A natural next step would be to apply these ideas in exploratory attacks or sensitivity analysis on sensory applications like speech recognition and natural language processing. To expand our procedural noise attack framework to other applications, it is worth identifying patterns in existing adversarial examples for domains like speech recognition [9, 11] and reinforcement learning [24, 37] to find analogues of procedural noise. As a starting point, these patterns can be found by finding perturbations that maximize the hidden layer difference as in the Singular Vector Attack [30] or by using feature visualization on earlier layers of DCNs to infer the low-level features that a model learns.

7 PRELIMINARY DEFENCE

The DCNs we tested are surprisingly fragile to procedural noise as UAPs. Improving their robustness to adversarial examples is not a straightforward task. A robust defence needs to defend not only against existing attacks but also future attacks, and often proposed defences are shown to fail against new or existing attacks [8].

Among existing defences, adversarial training appears to be more robust than others [1]. However, we have shown in Table 2 that ensemble adversarial training against gradient-based attacks did not significantly diminish the input-specific evasion rate of our procedural noise attack. This suggests that such defences do not generalize well as the attacks used for adversarial training do not sufficiently represent the entire space of adversarial examples. Training against all types of adversarial attacks would become computationally expensive, especially for high-dimensional tasks like ImageNet. Thus, defences that regularize the model or incorporate domain-specific knowledge may be more efficient strategies.

DCNs' weakness to adversarial perturbations across inputs may be a result of their sensitivity—small changes to the input cause large changes to the output. Thus, input-agnostic defences that minimize the impact of small perturbations may be effective in reducing the models' sensitivity without the need to train against all types of attacks. As a preliminary investigation, we briefly explore here using denoising to defend against the universality of our procedural noise perturbations.

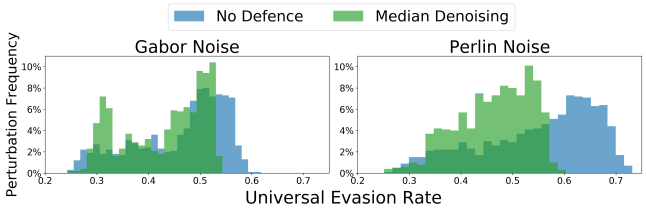

Figure 4: Histogram of metrics over all perturbations on Inception v3 with and without median denoising for (left) Gabor noise and (right) Perlin noise for ℓ_∞-norm $\varepsilon = 16$.

7.1 Denoising

Denoising with spatial filters is a common pre-processing step in signal processing and is thus an attractive defence due to its simplicity and pervasiveness across signal processing applications. Median filtering is a smoothing operation that replaces each entry with the median of its neighbouring entries, and it often preserves edges while removing noise. The idea is to smooth out the high-frequency noise, which we know to be highly correlated with the universal evasion rates for Perlin noise on Inception v3.

Experiment. We retrain Inception v3 with a median denoising filter applied as a pre-processing step. For this evaluation, we use the same 1,000 of each procedural noise, 1,000 uniform random perturbations, and 5,000 validation set points as in Sect. 4. We test how denoising improves robustness for ℓ_∞ attack norms $\varepsilon = 4, 8, 12$, and 16. The decrease in the number of uniform random perturbations is due to results in Sect. 4, where we found that its universal evasion rate had very low variance across 10,000 samples, and our results show it is similar for 1,000 samples.

Results. The min-max oscillations present in the procedural noise perturbations may have allowed the noise patterns to persist despite the denoising, as the universal evasion rates are not completely mitigated by the defence. Across the different ℓ_∞-norm values, the denoising decreased the median and mean universal evasion rates by a consistent amount when compared with no defence: 7.2-10.8% for Gabor noise and 13.2-16.9% for Perlin noise. Fig. 4 shows that the decrease in effectiveness of Perlin noise is much greater than that for Gabor noise, suggesting that the denoising appears to slightly mitigate the effectiveness of high-frequency noise patterns.

This denoising defence grants a reasonable increase in robustness despite the simplicity of the algorithm. However, this measure has not fully mitigated the sensitivity to procedural noise. We hope future work explore more input-agnostic defences that minimize the sensitivity of large-scale models to small perturbations with techniques like model compression [20], Jacobian regularization [27, 30, 60, 67], or other types of denoising.

8 CONCLUSION

We highlight the strengths of procedural noise as an indiscriminate and inexpensive black-box attack on DCNs. We have shown that popular DCN architectures for image classification have systemic vulnerabilities to procedural noise perturbations, with Perlin noise UAPs that achieve 58% or larger universal evasion across non-adversarially trained models. This weakness to procedural noise can be used to craft efficient universal and input-specific black-box attacks. Our procedural noise attack augmented with Bayesian optimization attains competitive input-specific success rates whilst improving the query efficiency of existing black-box methods over 100 times. Moreover, we show that procedural noise attacks also work against the YOLO v3 model for object detection, where it has an obfuscating effect on the "person" class.

Our results have notable implications. The universality of our black-box method introduces the possibility for large-scale attacks on DCN-based machine learning services, and the use of Bayesian optimization makes untargeted black-box attacks significantly more efficient. We hypothesize that our procedural noise attacks exploit low-level features that DCNs are sensitive towards. If true, this has worrying implications on the safety of transfer learning, as it is often these low-level features that are preserved when retraining models for new tasks. It may be the case that universal attacks

can be extended to other application domains by using compact representations of UAPs that exploit the sensitivity of models to low-level features.

We have shown the difficulty of defending against such novel approaches. In particular, ensemble adversarial training on gradient-based methods was unable to significantly diminish our procedural noise attack. We suggest that future defences take more input-agnostic approaches to avoid the costs in defending and retraining against all possible attacks. Our work prompts the need for further research of more intuitive and novel attack frameworks that use analogues of procedural noise in other machine learning application domains such as audio processing. We hope future work will explore in more depth the nature of cross-model universal adversarial perturbations, as these vulnerabilities generalize across both inputs and models, and a more formal framework could better explain why our procedural noise attacks are effective.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by the Data Spartan research grant DSRD201801.

REFERENCES

[1] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, Stockholmsmässan, Stockholm Sweden, 274–283.

[2] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. 2006. Can machine learning be secure?. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS '06)*. ACM, New York, NY, USA, 16–25. https://doi.org/10.1145/1128817.1128824

[3] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2018. Black-box attacks on deep neural networks via gradient estimation. In *ICLR Workshop*.

[4] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 387–402.

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[6] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248* (2017).

[7] Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599* (2010).

[8] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* (2019).

[9] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *USENIX Security Symposium*. 513–530.

[10] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Symposium on Security and Privacy*. 39–57.

[11] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944* (2018).

[12] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Workshop on Artificial Intelligence and Security*. 15–26.

[13] Yali Du, Meng Fang, Jinfeng Yi, Jun Cheng, and Dacheng Tao. 2018. Towards query efficient black-box attacks: An input-free perspective. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. ACM, 13–24.

[14] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2018. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231* (2018).

[15] Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[16] Ian Goodfellow. 2018. Defense against the dark arts: An overview of adversarial example security research and future research directions. *arXiv preprint arXiv:1806.04169* (2018).

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT Press.

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[19] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435* (2016).

[20] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

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

[22] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

[23] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. 2011. Adversarial machine learning. In *Workshop on Security and Artificial Intelligence*. 43–58.

[24] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[25] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. 2019. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*.

[26] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175* (2019).

[27] Daniel Jakubovitz and Raja Giryes. 2018. Improving DNN robustness to adversarial attacks using Jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 514–529.

[28] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. 2016. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT'15)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 21–26. https://doi.org/10.4108/eai.3-12-2015.2262516

[29] Min-Joo Kang and Je-Won Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one* 11, 6 (2016), e0155781.

[30] Valentin Khrulkov and Ivan Oseledets. 2018. Art of singular vectors and universal adversarial perturbations. In *Procs. Conf. on Computer Vision and Pattern Recognition*. 8562–8570.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[32] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[33] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *International Conference on Learning Representations*.

[34] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. 2010. A survey of procedural noise functions. In *Computer Graphics Forum*, Vol. 29. 2579–2600.

[35] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 54.

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[37] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748* (2017).

[38] Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1-3 (1989), 503–528.

[39] André Teixeira Lopes, Edilson de Aguiar, Alberto F De Souza, and Thiago Oliveira-Santos. 2017. Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order. *Pattern Recognition* 61 (2017), 610–628.

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks.

arXiv preprint arXiv:1706.06083 (2017).
[41] Patrick McDaniel, Nicolas Papernot, and Z Berkay Celik. 2016. Machine learning in adversarial settings. *IEEE Security & Privacy* 14, 3 (2016), 68–72.
[42] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. 2016. Sparse gaussian processes for Bayesian optimization. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence (UAI'16)*. AUAI Press, Arlington, Virginia, United States, 517–526.
[43] J Moćkus, V Tiesis, and A Źilinskas. 1978. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization* 2 (1978), 117–129.
[44] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Conference on Computer Vision and Pattern Recognition*. 86–94.
[45] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *Conference on Computer Vision and Pattern Recognition*. 2574–2582.
[46] Konda Mopuri, Utkarsh Ojha, Utsav Garg, and R. Venkatesh Babu. 2018. NAG: Network for adversary generation. In *Procs. Conf. on Computer Vision and Pattern Recognition*. 742–751.
[47] Fabrice Neyret and Eric Heitz. 2016. *Understanding and controlling contrast oscillations in stochastic texture algorithms using spectrum of variance.* Ph.D. Dissertation. LJK/Grenoble University-INRIA.
[48] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill* 2, 11 (2017).
[49] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1717–1724.
[50] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).
[51] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Asia Conference on Computer and Communications Security*. 506–519.
[52] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. SoK: Security and privacy in machine learning. In *European Symposium on Security and Privacy*. 399–414.
[53] Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.
[54] Ken Perlin. 2002. Improving noise. *ACM Transactions on Graphics* 21, 3 (2002), 681–682.
[55] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*. The MIT Press.
[56] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.
[57] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.
[58] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
[59] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.
[60] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, USA, 833–840.
[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.
[62] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware detection using two dimensional binary program features. In *Intl. Conference on Malicious and Unwanted Software (MALWARE)*. 11–20.
[63] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv preprint arXiv:1702.08568* (2017).
[64] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2016. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2016), 148–175.
[65] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[66] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural information Processing Systems*. 2951–2959.
[67] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. 2017. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing* 65, 16 (2017), 4265–4280.
[68] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2013. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3476–3483.
[69] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *AAAI*, Vol. 4. 12.
[70] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*. 2818–2826.
[71] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[72] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. ACM, 303–314.
[73] Florian Tramér, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*.
[74] Bichen Wu, Forrest N Iandola, Peter H Jin, and Kurt Keutzer. 2017. SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 446–454. https://doi.org/10.1109/CVPRW.2017.60
[75] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610* (2018).
[76] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in neural information processing systems*. 3320–3328.
[77] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. 2018. Transferable adversarial perturbations. In *Computer Vision–ECCV 2018*. Springer, 471–486.

A EXAMPLES OF PERTURBATIONS

Figs. 5 and 6 show examples of perturbations from procedural noise functions and existing white-box UAP attacks. Notice the visual similarities in structure between the noise patterns in Figs. 5 and 6.

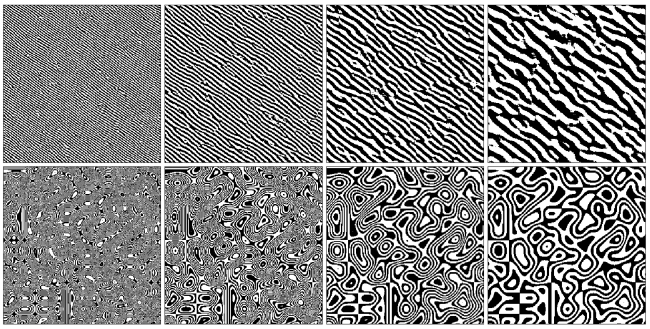

Figure 5: Procedural noise patterns with (top) Gabor noise and (bottom) Perlin noise, both with decreasing frequency from left to right.

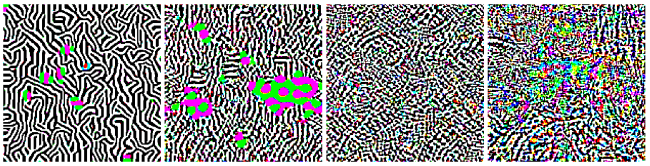

Figure 6: UAPs generated for VGG-19 using the white-box Singular Vector Attack [30].

B TOP 5 INPUT-SPECIFIC EVASION

In Table 5, we consider the top 5 input-specific evasion rate where the true class label is outside the top 5 class labels in the perturbed image $x+s$. The top 5 error is often used alongside the top 1 error to measure the performance of classifiers on ImageNet. Top 5 evasion is more difficult than top 1 as the confidence in the true label has to be degraded sufficiently enough for it to be below five other class labels. Despite that, Perlin noise is able to achieve top 5 evasion on more than half the inputs for all tested models. We see that our procedural noise remains a strong attack for the top 5 error metric.

Table 5: Top 5 input-specific evasion rate (in %) for random and procedural noise perturbations. Original refers to the top 5 error on the unaltered original images. The strongest attack on each classifier is highlighted.

Classifier	Original	Random	Gabor	Perlin
VGG-19	9.5	28.0	90.2	**92.6**
ResNet-50	8.4	29.5	82.5	**85.3**
Inception v3	6.3	18.0	66.9	**79.5**
IRv2	4.5	13.2	56.3	**66.4**
$IRv2_{ens}$	5.1	12.1	44.1	**53.6**

C GABOR NOISE RESULTS

The Gabor noise results for input-specific and universal black-box attacks in Sect. 5 against Inception v3 are in Tables 6 and 7. Gabor noise attacks are labeled with "gab".

Table 6: Results for Gabor noise input-specific black-box attacks on Inception v3 with ℓ_∞-norm $\varepsilon = 16$.

Attack	Query Limit	Average Queries	Success Rate (%)
$BayesOpt_{gab}$	100	9.8	83.1
$BayesOpt_{gab}$	1,000	10.9	83.6
$L\text{-}BFGS_{gab}$	100	6.0	44.7
$L\text{-}BFGS_{gab}$	1,000	6.0	44.7
$Random_{gab}$	1,000	62.2	86.1

Table 7: Results for Gabor noise universal black-box attacks on Inception v3 with ℓ_∞-norm $\varepsilon = 16$. Universal evasion rates (%) of the optimized perturbations are shown for their respective training set and the validation set.

	$BayesOpt_{gab}$		$L\text{-}BFGS_{gab}$	
Train Size	Train	Val.	Train	Val.
50	64.0	57.6	58.0	51.6
125	64.6	58.0	58.4	54.6
250	60.0	58.4	58.8	56.0
500	64.8	62.4	59.8	58.0

D CORRELATION MATRICES

Figs. 7 and 8 show the correlation matrices between the procedural noise parameters with the corresponding universal evasion rates on the various ImageNet classifiers and the performance metrics on YOLO v3 respectively. These correlations quantify the effects of each parameter and the transferability of UAPs across models. "V19", "R50", and "INv3" refer to VGG-19, ResNet-50, and Inception v3 respectively.

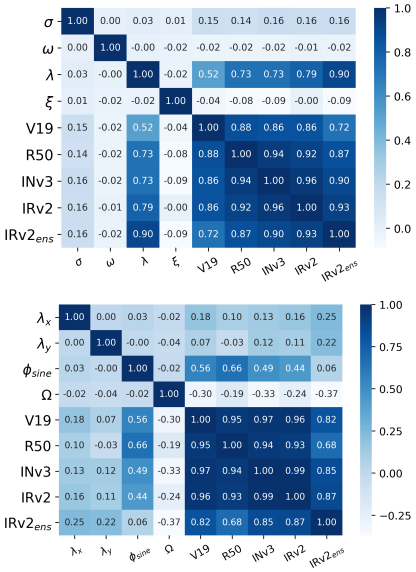

Figure 7: Correlations between universal evasion rates with (top) Gabor noise parameters ($\delta_{\mathbf{gab}} = \{\sigma, \omega, \lambda, \xi\}$) and (bottom) Perlin noise parameters ($\delta_{\mathbf{per}} = \{\lambda_x, \lambda_y, \phi_{\mathbf{sine}}, \omega\}$).

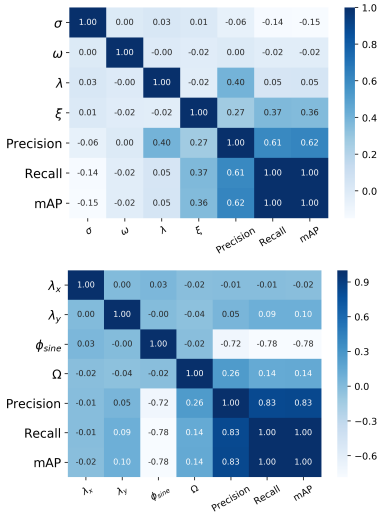

Figure 8: Correlations between YOLO v3 metrics with (top) Gabor noise parameters ($\delta_{\mathbf{gab}} = \{\sigma, \omega, \lambda, \xi\}$) and (bottom) Perlin noise parameters ($\delta_{\mathbf{per}} = \{\lambda_x, \lambda_y, \phi_{\mathbf{sine}}, \omega\}$).