LaVAN: Localized and Visible Adversarial Noise

Danny Karmon [1] **Daniel Zoran** [2] **Yoav Goldberg** [3]

Abstract

Most works on adversarial examples for deep-learning based image classifiers use noise that, while small, covers the entire image. We explore the case where the noise is allowed to be visible but confined to a small, localized patch of the image, without covering any of the main object(s) in the image. We show that it is possible to generate localized adversarial noises that cover only 2% of the pixels in the image, none of them over the main object, and that are transferable across images and locations, and successfully fool a state-of-the-art Inception v3 model with very high success rates.

Padlock (92.7%)

Tiger Cat (94.4%)

Car Mirror (94.5%)

Stingray (90.5%)

[1]Microsoft, Herzliya, Israel. [2]DeepMind, London, UK. [3]Department of Computer Science, Bar-Ilan University,Ramat Gan, Israel.. Correspondence to: Danny Karmon <danny.karmon@microsoft.com>, Yoav Goldberg <yogo@cs.biu.ac.il>, Daniel Zoran <danielzoran@google.com>.

Proceedings of the 35^{th} International Conference on Machine Learning,Stockholm, Sweden, PMLR 80, 2018.

1. Adversarial Noise

Deep neural-network architectures achieve remarkable results on image classification tasks. However, they are also susceptible to being fooled by adversarial examples: input instances which were modified in a particular way, and as a result, are misclassified by the network. Of course, for the adversarial example to be interesting, the change should be such that it does not confuse a human looking at the picture. Beyond the clear security implications, adversarial examples are also interesting as they may provide insights into the strengths, weaknesses, and blind-spots of these ubiquitous state-of-the-art classification models.

Most work on generating adversarial examples (we provide a more detailed review in section 5) focus either on noise which—while being imperceptible to humans—covers the entire image (Goodfellow et al., 2015; Szegedy et al., 2014), or on visible noise that covers prominent features of the main object in the image in a "natural" way (i.e., glasses with a specific pattern around a person's eyes in a face identification task (Sharif et al., 2016)). In contrast, we look at **visible noise** that is **localized** to a small area of the image (a bounded box with up to 2% of the pixels), and which **does not cover** the main object in the image. Figure 1 shows examples of such noised images that are misclassified by a state-of-the-art Inception V3 network with very high confidence.

A recent work by Brown et al (Brown et al., 2017) introduces a visible noise similar to ours. The works are complementary to a large extent. Their work focuses on the security implications and attempts to generate universal noise "patches" that can be physically printed and put on any image, in either a black-box (when the attacked network is unknown) or white-box (when the attacked network is known) setup. As a consequence, the resulting adversarial patches in (Brown et al., 2017) are relatively large (in a white-box setup, the generated noise has to cover about 10% of the image to be effective in about 90% of the tested conditions, and a disguised patch has to cover about 35% of the image for a similar result) and also visually resemble the target class to some extent. We do not attempt to produce a physical attack and are more interested in investigating the blind-spots of state-of-the-art image classifiers, and the kinds of noise that can cause them to misclassify.

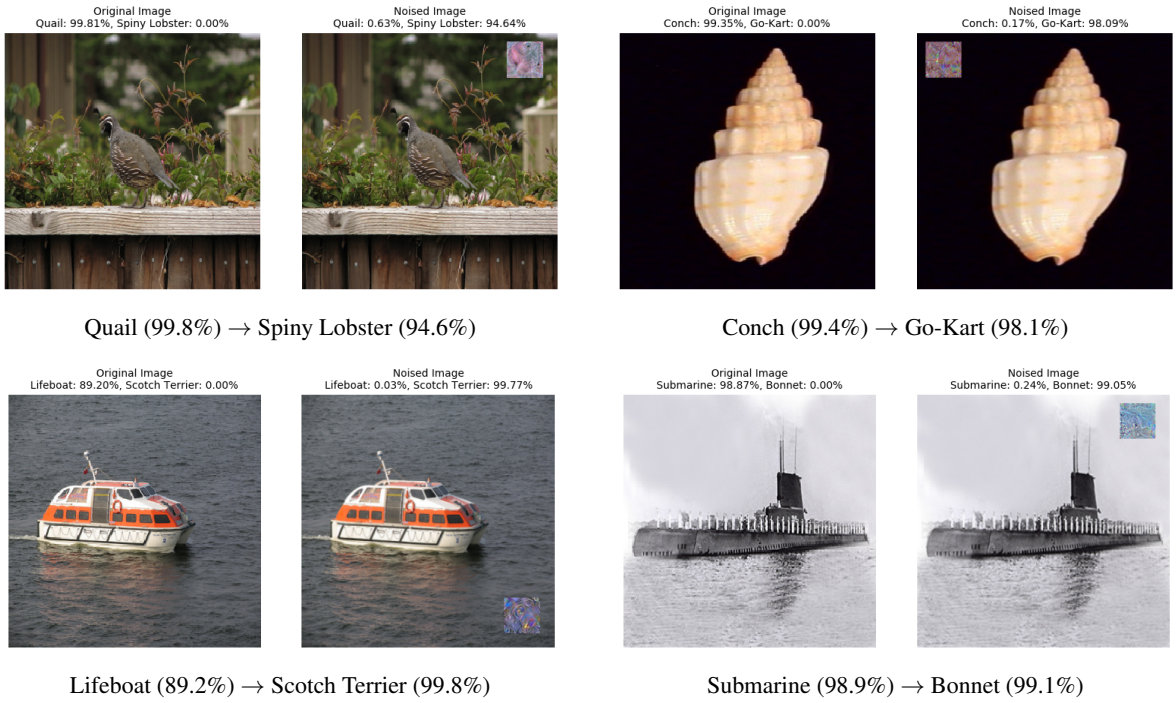


Figure 1. Images with network-domain localized noise. The Noise was generated for a specific input and location. Less than 2% of pixels are noised. Network-domain noises are scaled to image-domain for presentation.

We experiment with two setups: working in the *Network Domain* and in the *Image Domain*. In the network-domain case, the noise is allowed to take any value and is not restricted to the dynamic range of images. Such noise is akin to shining a very bright flash-light into someone's eye. In the image-domain case the noise is kept to the dynamic-range of images.

We show that in a white-box setting, we can generate localized visible noise that can be transferred to almost arbitrary images, covers only up to 2% of the image, does not cover any part of the main object in the image and yet manages to make the network misclassify with very high confidence. This works both for the network-domain and the image-domain case, although the success rates are naturally higher for the network-domain.

Moreover, by inspecting the gradients of the network over noised images, we show that the network does not usually identify the noised patch as the main cause of misclassification, and in some cases hardly assigns it any blame at all. The latter is true even in network-domain case. This is in contrast to the hypothesis posed in (Brown et al., 2017), in which the noise is said to be "much more salient" to the neural network than real-world objects.

The localized noises we generate are universal in the sense that they can be applied to many different images and locations. However, they are specific to a model they were trained on (i.e., equivalent to the white-box setups in (Brown et al., 2017)).

We believe these results highlight an interesting blind-spot in current state-of-the-art network architectures.

2. Localized noise for a single image and location

In the first setup, we explore generating a visible but localized adversarial noise that is specific to a *single image* and *location* within this image.

2.1. Setting and Method

Our method mostly follows that standard adversarial noise generation setup: we assume access to a trained model M that assigns membership probabilities $p_M(y|x)$ to input images $x \in R^{n=w\times h\times c}$. We denote by $\vec{y} = p_M(x)$ the vector of all class probabilities, and by $y = \arg\max_{y'} p_M(y = y'|x)$ the highest scoring class for input x (the classifier's prediction). Let y_{source} be the classifier's prediction on input x (the *source class*). We seek an image x' that is classified by the network as y_{target} (the *target class*). The image x' is composed of the original image with an additive noise $\delta \in R^n$: $x' = x + \delta$.

This is cast an optimization problem, seeking a value δ to to maximize $p_M(y = y_{\text{target}}|x + \delta)$. The noise δ can be found

using a stochastic gradient based algorithm.

We depart from this standard methodology by:

1. We want the noise δ to be confined to a small area over the image x, and to replace this area rather than be added to it. This is achieved by setting a mask $m \in \{0,1\}^n$, and taking the noised image to be $(1-m) \odot x - m \odot \delta$, where $\odot$ is element-wise multiplication.

2. Instead of training the noise to either maximize the probability of the target class or to minimize the probability of any other class (including the source class), we use a loss that does both things—it attempts to move the prediction towards the target class and away from the highest scored class. We use the network activations prior to the final softmax layer, denoted as $M(x)$, $M(y = y'|x)$. This decouples the outputs for the different classes, and speeds up convergence.

$$\arg\max_{\delta} [M(y = y_{\text{target}}|(1-m) \odot x + m \odot \delta) - M(y = y_{\text{source}}|(1-m) \odot x + m \odot \delta)] \tag{1}$$

Network vs Image domain Pixels values in each channel can take byte values (0-255), which are scaled floats in the range [0,1] when used as input to the network. We experiment with two noising setups: *network domain* and *image domain*. In the *network domain* case, the noise values is not restricted, and can go beyond the [0,1] dynamic range of the network. In the *image domain* case, the noise is restricted to be within 0 and 1. To facilitate image domain noises, we clip the noise patch to the range [0,1] after each gradient step.

The entire process is detailed in algorithm 1.

2.2. Experiments and Results

We use the PyTorch provided pre-trained Inception V3 network (Szegedy et al., 2016) trained on ImageNet. We use images of size 299x299 and noise a square patch of size 42x42, roughly 2% of the image pixels. We generate network-domain noise this way for 100 random (image, target-class, location) triplets. The location is chosen around the corners (we focus on the corners as they have the most chance of not covering the main object of the image). We consider the noising process successful if the network classifies the noised image to the desired target class with a confidence of at least 90%. For each image, we train the noise until we reach the desired confidence, or up to 10,000 iterations.

We managed to successfully generate localized network-domain noise this way to 79% of the 110 configurations we tried. When relaxing the requirement such that the network chooses the target class as the argmax (but perhaps with less than 90% probability), we manage to noise 91% of the configurations. By further relaxing the criteria to classify to any class other than the source one, noising success reaches 98%. Figure 1 shows examples of such noised images. Because network-domain noises are outside of the the range of valid pixels, we normalize the displayed noise patch to image range for visualization purposes.

2.3. Shortcomings

While we managed to produce a high confidence, targeted localized noise to 75 out of 100 (image, target-class, location) triplets we tried, the resulting noise is highly dependent on the exact image and location: attempts to shift the noise even a single pixel to either direction results in the network classifying to the original class with high confidence. Similarly, attempts to move the noise to a different image fail, unless we transfer, together with the noise, also a border of about 25% pixels to each side of the source image. In the next section, we generate noise that can be moved across images and locations.

3. Transferable localized noise

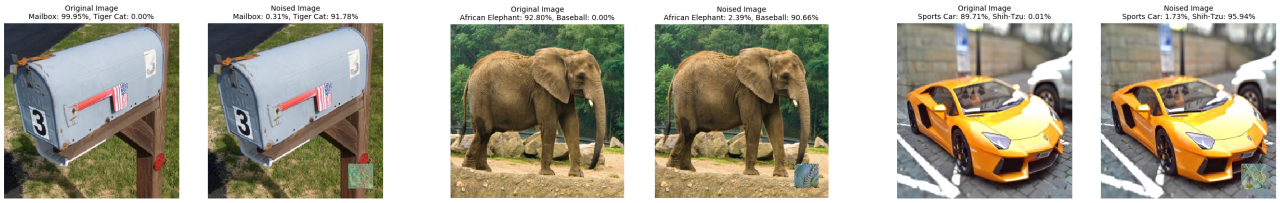
We now explore transferable ("universal") localized noises: a noise patch which can be applied across different images and image positions.

3.1. Method

We extend the localized noising process in Algorithm 1 by choosing, at each iteration, a random image x from a "training set" of 100 images, and a random location. We adjust

Algorithm 1 Localized Noising Process

Input: image x, model p_M, target class y_{target}, target probability s, mask m, noise domain d.
$y_{\text{source}} = \arg\max_y p_M(y|x)$
$\delta = \vec{0}$
$x' = (1-m) \odot x + m \odot \text{noise}$
repeat
  $\vec{y} = p_M(x')$
  $L_{\text{target}} = \ell(\vec{y}, y_{\text{target}}) = M(y = y_{\text{target}}|x')$
  $L_{\text{argmax}} = \ell(\vec{y}, y_{\text{argmax}}) = M(y = y_{\text{argmax}}|x')$
  $\nabla_{\text{target}} = \frac{\partial L_{\text{target}}}{\partial x}$
  $\nabla_{\text{argmax}} = \frac{\partial L_{\text{argmax}}}{\partial x}$
  $\delta = \delta - \epsilon \cdot (\nabla_{\text{target}} - \nabla_{\text{argmax}})$
  if $d =$*image domain* **then**
    $\delta = clip(\delta, 0, 1)$
  end if
  $x' = (1-m) \odot x + m \odot \delta$
until $p_M(y = y_{target}|x') \geq s$



Mailbox (99.9%) → Tiger Cat (91.8%) African-Elephant (92.8%) → Baseball (90.7%) Sports Car (92.8%) → Shih-Tzu (90.7%)

Daisy (95.6%) → **Tree Frog** (96.4%) Brown Bear (87.9%) → **Tree Frog** (82.7%) Minivan (90.7%) → **Tree Frog** (86.4%)

Figure 2. Transferable localized noises (network domain). Different network-domain noise patches, any of them can work on different images and locations. Notice that the same patch was used for all three images in the second row. The noises are scaled to image domain for visualization purposes.

Gong (89.3%) → **Jaguar** (96.4%) Mantis (96.8%) → **Cock** (71.8%) Brown Bear (92.5%) → **Baseball** (96.4%)

Figure 3. Transferable localized noises (image domain).

the noise vector and the mask so that they correspond to the target location, apply the noise to the image, and take a gradient step over the noise away from the source class of x and towards the shared target class y_{target}. Thus, at each iteration the same noise is applied on a random image and location, both sampled separately from uniform distributions of the possible images and locations respectively. This is very similar to the algorithm presented in (Brown et al., 2017), with a somewhat different loss function, as described above. We stop the noise generation process after the prediction model misclassifies with the desired confidence (i.e. Target class probability ≥ 0.9) for 30 consecutive iterations.

3.2. Experiments and Results

Overall we generated transferable noise patches for 14 different targets. The generated noises are presented in Figure 4, both for image domain noise patches and network domain patches (rescaled to image range for visualization purposes).

Network Domain We first provide anecdotal results of applying a trained noise to unseen images: Figure 2 shows some examples of images with transferable noises in the network domain.

To assess the robustness to location, we applied a noise patch striding across every second pixel in an image that did not participate in the noise generation process, evaluating the model's prediction at each location using the following metrics: (1) probability of predicting the source class; (2) probability of predicting the target class; (3) argmax indication (i.e whether the class received the highest probability among other class) for the target class, source class, or neither classes. Figure 5 shows the result of this process for the (image:Gondola, target:Volcano) pair.

As can be seen, the noise is very robust, reducing the source class probability to near 0 across the image, and similarly increasing the target class probability.

We repeated this process for each of the 14 targets and each of the 100 images in our test set. On average, applying the transferable noise to 83% of the possible locations caused the target class to receive a score ≥ 0.9, and in 97% of the locations it caused the model to not predict the original-source class.

To evaluate the noise's transferability across images, we

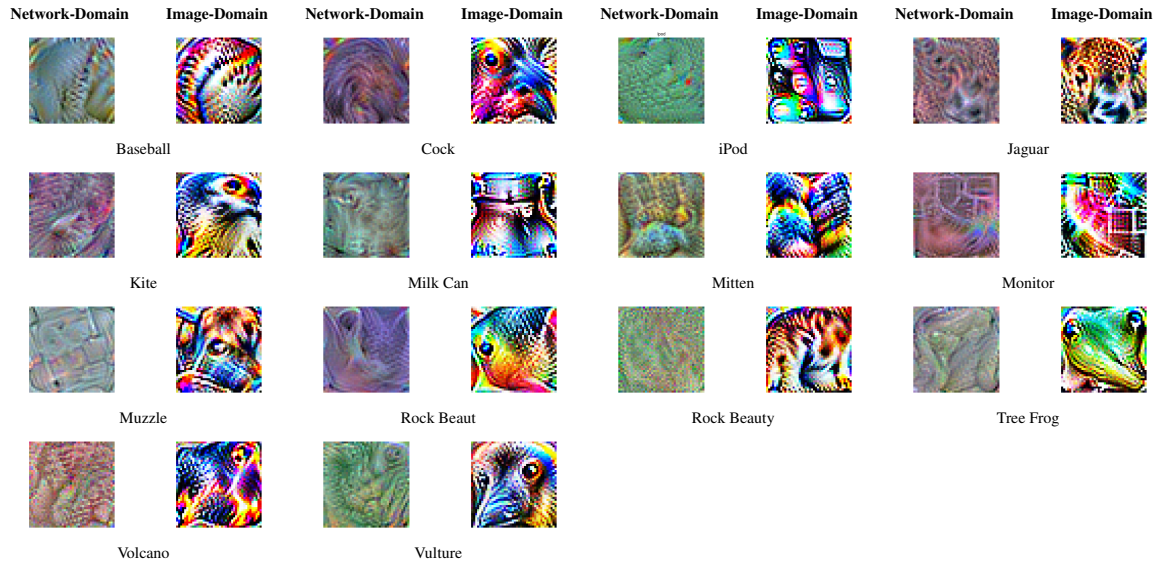


Figure 4. Network and image domain transferable localized noises for different target classes. Network domain noises were rescaled to image range for display purposes. Note that for many of the classes the resulting noises, and in particular the image-domain ones, contain prominent features of the target class like fur, eyes and global shape, and resembles a miniature version of the target class to some extent.

used a separate test set, consisting of 100 images from the ImageNet data-set that did not participate in the noise generation process. We placed each transferable noise patch on each image on the bottom right corner. Though only 43% of our attempts made the model predict the target class with confidence ≥ 0.9, in 89% of the cases it predicted the desired target as the most likely one, and in 100% of the cases, the transferable noise patches prevented the model from predicting the original source class.

Image Domain We evaluate the performance of the image domain noises using the same 100 images used in the network-domain evaluation. The success rates are somewhat lower, but still effective: while only 28.3% of the attempts made the model predict the target class with confidence ≥ 0.9, in 74.1% of the cases it caused the model to predict the target class as the most likely one, and in 78.9% it resulted in a misclassification of the source class. Figure 3 shows some examples of transferable image-domain noises.[1]

Class dependence We now ask the following question: are specific classes better at serving as "target" classes, making it easier to fool the network when using them? Conversely we ask, are particular source classes harder such that attempting to apply noise patches them results in less success?

Figure 8 summarizes the dependence of the noising process success on the source and target class using *image domain* noises. Each cell shows the success rate of the noising process when using source images from "source" class and using "target" as the target. As can be seen there are significant differences between the different classes. Since we only cover a small subset of all possible classes in ImageNet we can only postulate that more globally structured classes such as "volcano" are more challenging to this localized process than locally structured classes as "baseball".

4. Noise perceived by the network

We successfully generated small transferable, localized noise patches that fool a state-of-the-art network into misclassifying an example.

In (Brown et al., 2017), the authors suggest that the localized noise works because it is much more salient than natural looking objects in the scene, capturing all of the network's attention. To try and examine this claim, we take some noised images and attempt to "fix" the effect of the noise on the classification output.

First, we attempt to fix a noised image by feeding it into the network and taking gradient steps over the entire image towards the original source class.[2] For example, we take a noised image of a `Killer Whale` classified as a `Baseball`, and take gradient steps over the entire image until it is classified as a `Killer Whale`. The top image

[1] The image-domain patches resemble the target class to some extent. As a control, we also experimented with finding natural images that highly activate a target class, scaling them to 42x42 and pasting on the image. This fails completely. Similarly, generating highly-activating patches from white noise independently of any image, and then pasting them on images, also fails.

[2] This is similar to the salience map computation in (Simonyan et al., 2013), but we accumulate the absolute values of several gradient steps, until reaching the desired outcome.

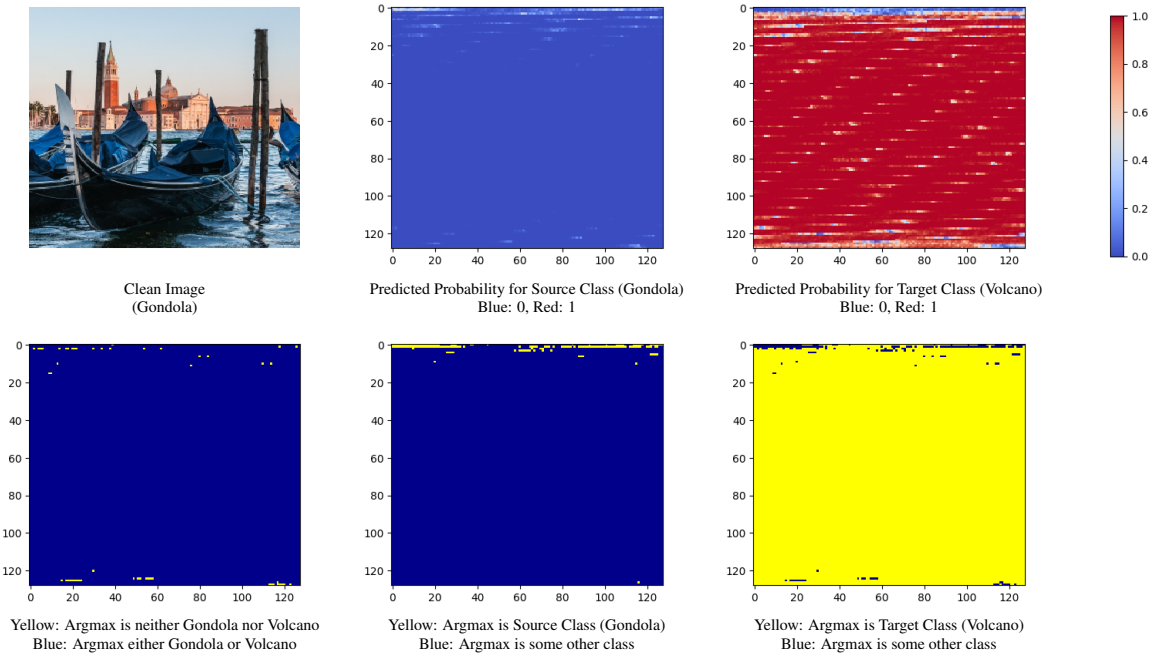


Figure 5. Result of placing the Volcano network-domain noise patch on different locations in the Gondola image

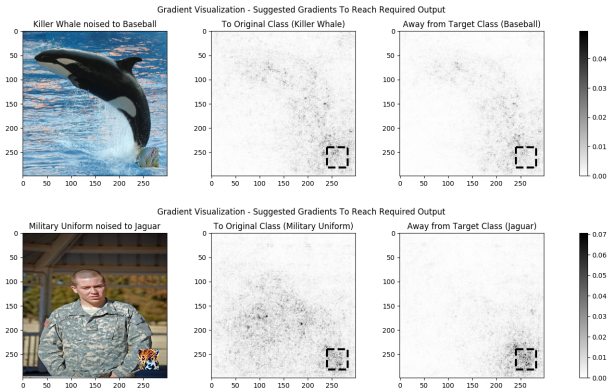


Figure 6. Gradient updates required to fix the `Baseball` (86%) (Top, Network Domain) and `Jaguar` (82%) (Bottom, Image Domain) misclassifications back to `Dolphin` and `Uniform`. In both cases, the noised area in both cases is well represented in the gradients, but is not the most salient area.

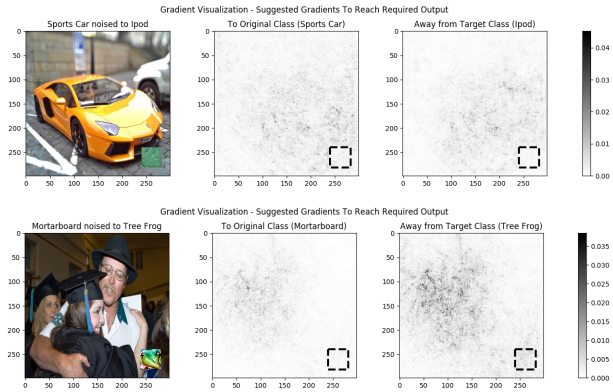


Figure 7. Gradient updates required to fix the `iPod` (98%) (Top, Network Domain) and the `Tree-frog` (80%) (Bottom, Image Domain) misclassifications. Noised area in both cases is mostly ignored in the gradients.

triplet in Figure 6 visualizes the gradient fix for the whale image. The center image shows the absolute values of the cumulative updates that were needed to overcome the noise and re-classify as a Whale. Notice that while a lot of the gradient efforts are within the noise patch, there is also a lot of activity *outside* of the patch, accenting features of the Whale's body.

What if instead of attempting to optimize towards the source class, we optimize the noised image *away from* the target class? I.e., rather than pushing the noised image to "be a whale again", we push it to "stop being a baseball"? The resulting cumulative gradients are displayed on the top images in Figure 6 (whale image triplet). We see less gradient activity overall, and, somewhat surprisingly, there again seems to be at least as much activity outside the noise patch as it is within it. This gradient behavior persists on different image/noise cases, with a combination of gradient within the noise box and outside of is.

Some noise cases are more intriguing. For example, the

Source Class \ Target Class	baseball	cock	iPod	jaguar	kite	milk-can	mitten	monitor	muzzle	rock beauty	tiger-cat	tree-frog	volcano	vulture
academic gown	0.18	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
banana	0.98	0.70	0.91	0.94	0.91	0.29	0.96	0.32	0.99	0.72	0.71	0.96	0.06	0.91
barbershop	0.99	0.92	0.34	0.93	0.89	0.63	0.91	0.50	0.98	0.69	0.78	0.95	0.22	0.89
bee	0.99	0.79	0.81	0.88	0.85	0.20	0.71	0.12	0.93	0.70	0.68	0.96	0.12	0.91
bell pepper	0.94	0.48	0.19	0.73	0.66	0.50	0.53	0.04	0.12	0.53	0.61	0.89	0.05	0.90
garden spider	0.97	0.66	0.10	0.95	0.89	0.46	0.19	0.22	0.99	0.84	0.83	0.96	0.05	0.71
book jacket	0.99	0.77	0.15	0.81	0.90	0.83	0.45	0.27	0.99	0.90	0.84	0.97	0.08	0.96
brown bear	0.80	0.21	0.13	0.30	0.32	0.31	0.21	0.02	0.50	0.19	0.32	0.55	0.01	0.28
bullet train	0.98	0.47	0.01	0.79	0.61	0.54	0.65	0.20	0.69	0.40	0.74	0.88	0.05	0.49
cardigan	0.94	0.51	0.25	0.77	0.66	0.18	0.55	0.17	0.93	0.29	0.56	0.71	0.15	0.45
corn	0.99	0.76	0.63	0.96	0.97	0.89	0.90	0.05	1.00	0.81	0.72	0.95	0.33	0.98
crash helmet	0.98	0.50	0.10	0.79	0.80	0.22	0.41	0.05	0.58	0.46	0.58	0.92	0.02	0.41
daisy	0.96	0.51	0.39	0.77	0.73	0.71	0.66	0.33	0.87	0.62	0.65	0.90	0.13	0.72
dishrag	0.97	0.67	0.06	0.90	0.95	0.52	0.63	0.02	0.95	0.67	0.73	0.92	0.10	0.92
dogsled	0.93	0.44	0.14	0.48	0.62	0.60	0.25	0.02	0.49	0.27	0.44	0.70	0.07	0.32
electric ray	0.98	0.77	0.15	0.92	0.94	0.98	0.89	0.14	0.99	0.84	0.73	0.95	0.13	0.98
football helmet	0.96	0.61	0.01	0.65	0.77	0.00	0.05	0.01	0.16	0.41	0.50	0.84	0.00	0.66
gong	0.95	0.36	0.03	0.84	0.71	0.11	0.42	0.02	0.50	0.28	0.72	0.75	0.01	0.37
green mamba	0.92	0.23	0.11	0.56	0.55	0.35	0.25	0.10	0.81	0.35	0.53	0.43	0.04	0.36
hatchet	0.50	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.07	0.00	0.00
hip	0.98	0.65	0.41	0.95	0.82	0.95	0.96	0.29	0.99	0.80	0.79	0.83	0.20	0.59
hummingbird	1.00	0.61	0.66	0.94	0.91	0.91	0.69	0.57	0.96	0.92	0.86	0.97	0.19	0.94
Indian cobra	0.96	0.29	0.30	0.76	0.64	0.36	0.14	0.16	0.87	0.60	0.59	0.77	0.05	0.64
iPod	0.99	0.79	1.00	0.96	0.88	0.10	0.45	0.10	0.94	0.80	0.89	0.93	0.13	0.94
jellyfish	0.99	0.52	0.75	0.91	0.87	0.93	0.56	0.31	0.96	0.84	0.89	0.94	0.05	0.89
lacewing	0.93	0.33	0.05	0.72	0.73	0.20	0.24	0.02	0.90	0.17	0.36	0.46	0.05	0.15
mailbag	0.84	0.23	0.02	0.44	0.60	0.13	0.10	0.00	0.69	0.02	0.68	0.73	0.02	0.05
mantis	0.94	0.47	0.27	0.71	0.66	0.27	0.35	0.19	0.59	0.39	0.67	0.73	0.10	0.59
military uniform	0.97	0.82	0.15	0.90	0.87	0.69	0.90	0.33	0.97	0.68	0.67	0.90	0.21	0.82
minivan	0.98	0.87	0.13	0.92	0.91	0.76	0.95	0.09	1.00	0.78	0.80	0.97	0.09	0.92
mortarboard	0.97	0.48	0.24	0.60	0.66	0.10	0.55	0.07	0.66	0.36	0.51	0.67	0.06	0.56
poncho	0.99	0.89	0.51	0.92	0.93	0.37	0.93	0.18	1.00	0.78	0.88	0.95	0.05	0.92
rapeseed	0.99	0.71	0.51	0.93	0.90	0.91	0.96	0.27	1.00	0.86	0.75	0.94	0.38	0.83
starfish	0.99	0.82	0.69	0.95	0.77	0.50	0.86	0.17	0.99	0.82	0.74	0.98	0.07	0.95
turnstile	0.99	0.78	0.01	0.93	0.74	0.50	0.56	0.02	0.96	0.23	0.72	0.94	0.05	0.73
velvet	0.98	0.66	0.19	0.91	0.90	0.61	0.86	0.04	1.00	0.63	0.49	0.92	0.11	0.82
walking stick	0.97	0.40	0.54	0.79	0.78	0.54	0.54	0.27	0.67	0.59	0.76	0.84	0.10	0.69
washer	0.98	0.64	0.17	0.90	0.78	0.51	0.78	0.25	0.93	0.61	0.80	0.93	0.07	0.66
water snake	0.99	0.75	0.59	0.94	0.97	0.94	0.86	0.22	1.00	0.89	0.69	0.95	0.28	0.96
wing	0.99	0.81	0.50	0.93	0.99	0.96	0.98	0.27	0.96	0.94	0.74	0.93	0.49	0.99

Figure 8. Average target score heatmap for transferable image-domain noises. The value in $C_{i,j}$ is the average score assigned to the target class j when that the source class is i.

top image triplet in Figure 7 shows the case of a `Sports Car` being noised to a `iPod` at 98%. When driving the image towards `Sports Car` (center), the gradient seem to accent the car's front and its surrounding, while *almost completely ignoring* the noised patch. In order to restore the original class, the network's gradients work to enhance features of the original class rather than diminishing the intruding noise. This pattern persists when optimizing the noised image *away from* `iPod` (right): there are hardly any updates within the noised area. The stealthiness of the noise is observed both in network-domain noises (Fig 7 top) and in image-domain noises (Fig 7 bottom).

This suggests that, at least in some image/noise combinations, the localized visible noise is not only very effective at misleading the classifier and triggering the target class, it is also quite *stealthy*–at least according to the target gradients wrt to the image, the classifier attributes the target to various elements in the image, but not to the noisy patch itself. This is contrast to previous suggestions that noise is somehow more salient to the network than other parts of the image.

4.1. Quantifying the saliency of noise

To further quantify the above behavior, we consider the gradients required to fix the 2,800 noised images that result from applying each of the 28 noises in Figure 4 to the bottom right corner of each of the 100 test images.

In each gradient-fix map, we score each 42x42 patch according to: (1) the maximum absolute value within the patch (MAX); and (2) the sum of the absolute values within the patch (SUM). We then take the patch with the maximum value according to each metric, and check if it overlaps with the noise location. Table 1 lists the results. According to both metrics, the most active patch very rarely overlaps with the noise location. This strengthens our previous claim and suggests that the model may fail to understand and locate the noise in the image.

Domain	Fix method	MAX	SUM
Network	Towards Source	0.4%	0.15%
Network	Away from Target	0.5%	0.13%
Image	Towards Source	0.6%	5.4%
Image	Away from Target	0.7&	5.2%

Table 1. Quantifying the saliency of transferable noises. Each cell lists the percent of out of 1,400 (images,target) pairs in which the most active 42x42 gradient-fix patch ovelaps with the 42x42 noise patch. MAX: active patches are selected according to the max value in the patch. SUM: selected according to sum of values in the patch.

5. Related Work

Most works focus on adding a small amount of noise, imperceptible to the human eye, but covering the entire image. It was shown that it is possible to construct such noises to thwart multiclass image classifiers (Goodfellow et al., 2015; Szegedy et al., 2014; Moosavi-Dezfooli et al., 2016), and, more recently, also structured classifiers in tasks such as image segmentation and pose estimation (Cisse et al., 2017). In all these cases the noise covers the entire image, including the salient objects within it. We focus on cases where the noise may be larger, but only affect a small part of the image.

Several works explore setups that do not noise the entire image. Sharif et al (Sharif et al., 2016) causes a face recognition system to identify an incorrect face by adding glasses with a specially constructed frame texture, and Evtimov et al (Evtimov et al., 2017) causes misclassification of traffic signs by adding a specific pattern of rectangular, solid-colored patches on top of a traffic sign. These are very impressive works in which the attack transfers to images taken in the real world. However, from the noise locality perspective, in both these cases, the noise patches attack very prominent characteristics of the objects to be classified. The glasses hide areas around the eyes, which are very indicative for face recognition. Similarly, many traffic signs can be abstracted as a specific arrangement of solid-colored rectangles on a solid-colored background, so it is not too surprising that messing with the rectangles formation can fool a traffic-sign classifier. Su et al (Su et al., 2017) demonstrate that, in about 70% of 32x32 images from CIFAR-10, it is possible to find a single pixel whose change in value can cause a misclassification (the number drops to between 20% and 30% for a targeted attack towards a specific target class). However, this requires a very small image size, and the offending pixel usually at the center of the classified image. Papernot et al (Papernot et al., 2016) show that, when considering black-and-white digits classification, changing a small percentage of pixels from black to white can cause misclassification. The pixels are spread out across the entire image and look as harming noise to a human observer.

In contrast to all these works, in our case, the pixels are localized to a specific region of the image and do not cover the source-class object at all. Finally, the recent *adversarial patch* work by Brown et al (Brown et al., 2017) is very similar to our setting, and was discussed throughout this report. They focus on printable patches that are immune to scaling or ration, and that can be used for physical attacks. As a consequence, their patches are required to be very large, covering upward from 10% of the image in order to be effective. We demonstrate that, when relaxing these requirements, networks can be fooled also by much smaller patches of visible noise, that cover a substantially smaller area of the image. Moreover, we show that—at least according to its gradients—the network does not recognize the attacking patches as the source of the misclassification.

6. Conclusions

We show that it is possible to learn visible and localized adversarial noise patches that cover only 2% of the pixels in the image, none of them on the main salient object, and that cause a state-of-the-art image classifier to misclassify to arbitrary labels. Such noise patches are not image specific, and the same patch can be applied to arbitrary images and locations, causing a misclassification to the desired target class with very high success rates. Perhaps surprisingly, in many cases, the noise patch is not particularly salient as far as the network is concerned as revealed by the resulting gradients when one attempts to "fix" the noise effect.

The resulting noise patches resemble the target classes both in the network domain and image domain cases. Prominent features such as texture and fur, body parts like eyes and beaks and global shape characteristics are all reflected in the outputs. The noise patch—however similar to the target class—is still much smaller than main object in the image, yet makes the network misclassify.

Furthermore, and somewhat surprisingly when considering that the input patch lies pretty close to stimuli the network usually observes (certainly in the image domain case), the gradients when attempting to revert the noise tell us that the noised patch is not very salient to the network when compared to other locations within the image.

Considering the noised patch is not salient to the network raises some concerns from a security perspective, but also raises some interesting questions about the inner workings of the network. From a security perspective, such a noise patch is problematic as it seems that the network does not "see" that is being fooled, possibly vulnerable for such attacks. This is true both for the image and network domain noise. From a scientific perspective we feel the methods and results presented may serve as the foundation of some interesting future research. The brittleness of the classification output of these networks tells us something about the way they operate. One could ask whether different architecture are susceptible to such attacks in different ways and whether specific architectural choices affect the resulting vulnerabilities. These are all left for future research.

Acknowledgements

The work at Bar-Ilan University was supported by The Israeli Science Foundation (grant number 1555/15), Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI) and The Allen Institute for Artificial Intelligence.

References

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Cisse, M. M., Adi, Y., Neverova, N., and Keshet, J. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6980–6990. Curran Associates, Inc., 2017.

Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017. URL http://arxiv.org/abs/1707.08945.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016. URL http://arxiv.org/abs/1610.08401.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, 2016.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1528–1540. ACM, 2016.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL http://arxiv.org/abs/1312.6034.

Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017. URL http://arxiv.org/abs/1710.08864.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2014.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.