An infinity norm constrained attack for the NIPS 2017 Competition Track

Nozdryn-Plotnicki, Aleksey
Wightman, Ross

Abstract

We present our attacks for the NIPS 2017 Adversarial Attacks and Defenses competition. We placed 5th (Targeted) and 9th (Non-targeted). Images were perturbed within an infinity norm limit in order to fool black box classifiers submitted by other competitors. Our iterative attack makes a change of variable in order to perform unconstrained optimization with Adam. We introduce differentiable augmentations and resizes in order to defeat simple augmentation-based defenses and to attack ensembles models with different input sizes. We pre-compute single, targeted, input-agnostic perturbations that transfer well between models.

1 Introduction

Surprisingly, perhaps, it is possible to manipulate the pixels of a natural image in such a way that a deep neural network classifier is fooled into predicting the wrong class, when a human would consider the true class of the image to be unchanged. This gives rise to concerns that deep neural networks deployed in the real world could be vulnerable to attacks from adversaries. A self-driving car that misinterprets its environment could have tragic consequences. Thus there is great interest in establishing what can be done in terms of hardening classifiers to be resistant to such attacks, and to establish what are the strongest attacks.

It is perhaps less surprising that a white box attack is possible. With full access to a model and its weights, it is possible to craft very small perturbations to a natural image in order to fool a deep neural network. It is more surprising that it is possible attack a known model in a white box manner, but then have that image also fool a different neural network with a different architecture and different weights. This is the most likely real-world scenario.

In August and September, 2017, Kaggle and Google Brain hosted a trio of competitions for the NIPS 2017 Competition Track, "Adversarial Attacks and Defenses". The competitions created an interesting adversarial images environment:

- Black box: Target classifiers being attacked were truly black box, not artificially black box as in a typical research paper.
- Defenses were highly engineered: Rather than pure or idealized, competitors were free to do whatever works
- Limited computation: Approaches needed to be computationally efficient, a likely constraint in any real-world application

1.1 Competition basics

In the "Targeted Adversarial Attack" competition, the inputs are sets of 100 images and 100 target classes. Attacks are given 500 seconds per set to create 100 adversarial images that will cause a classifier to output the given target class. In the "Non-targeted Adversarial Attack" competition, the situation is the same, except the goal is simply for the classifier to output the incorrect class rather than the correct one. For each batch of 100 images, an infinity-norm max epsilon is selected randomly out of 4, 8, 12, or 16 (in [0,255] pixel space). Any output images from attacks will be clipped so that the infinity-norm of the difference between the original image and the attacked image is less than or equal to the max epsilon.

In the "Defense Against Adversarial Attack" competition, the inputs are sets of 100 images that have been attacked in the other two tracks. Defenses are given 500 seconds to output a set of 100 classes for each batch.

All attacks are run against all sets of 100 images. All defenses are run against all attacked sets. Defenses score a point for every image they correctly classify. Targeted attacks score a point for every time a defense outputs the target class. Non-targeted attacks score a point for every time a defense outputs the incorrect class. The goal is to maximize score.

Attacks and defenses were run on a n1-highmem-4 Google Cloud Platform instance with one K80 GPU attached to it. This is the environment in which the time limit is enforced.

1.2 Motivation

We sought to create attacks that were robust against simple defensive augmentations.

In the first round of the competition we, independently and un-merged, were very successful using simple augmentation for defense. We placed 1st and 2nd with normalized scores of 89.9% and 87.2%, surpassing even the baseline adv inception v3 model [4]. All with only naturally trained models.

Our best implementation in tensorflow took 5 crops, doubled that by mirroring, and doubled that again by blurring with a Gaussian blur radius 1.5. All 20 samples were evaluated with the Inception Resnet v2 [2] model and ensembled with a simple mean of the probability.

The time/compute constraint in the competition was somewhat tight, so we could benefit greatly for anything that could be pre-computed with unbounded compute. In our case, the universal perturbations take that opportunity.

2 Our iterative attack

Our attack is in the same gradient-based iterative family with the Fast Gradient Sign Method (FGSM) [5] and projected gradient descent (PGD) [6] attacks. We optimize a loss function by changing a perturbation to the image. Our attack performs a change of variable in order to do unconstrained gradient descent.

We perform black-box attacks by building white box attacks against substitute defenses, "target models", that we do have access to, seeking to maximize transferability as we do so. Ironically, in the case of the competition, most defenses were in fact using models that were available publicly, so truly a mix of both was happening.

We maximize transferability by minimizing the loss, introducing augmentations to the target model, and by attacking as large and diverse an ensemble as possible.

2.1 Loss function

We seek to minimize the cross-entropy loss of the perturbed image for a target model, for some target class. In the case of a targeted attack, this is obviously the target.

For non-targeted attacks we have several choices:

- Maximize the cross-entropy loss for the class predicted by the target model for the unperturbed image
- Minimize the cross-entropy loss based on the outputs of the target model for the unperturbed image:
  - Least likely class
  - Random class
  - nth most likely class

2.2 Box constraints

In principle we face two constraints. The perturbed image must remain a valid image and the perturbation is constrained by the infinity norm.

$$x + \delta \in [0,1]^n$$
$$\delta \in [-\epsilon, \epsilon]^n$$

We can re-write this as a single constraint for each pixel value i:

$$-min(x_i, \epsilon) \leq \delta_i \leq min(1 - x_i, \epsilon)$$

We employ a change of variable w:

$$\delta_i(w_i, x_i) = \begin{cases} min(x_i, \epsilon) \cdot tanh(w_i) & w_i \leq 0 \\ min(1 - x_i, \epsilon) \cdot tanh(w_i) & w_i \geq 0 \end{cases}$$
$$w_i \in [-\infty, \infty]$$

This then allows us to perform an unconstrained gradient descent.

2.3 Augmentations

Simple image augmentations have been shown to dramatically foil black box attacks, and indeed we confirmed this in the first round of the competition. A blur [7] and random crops/foveation [8] were very effective. We seek to create adversarial examples that are robust to this defense. To do so, we perform differentiable augmentations after perturbing the image and before passing to the target model. This allows us to compute derivatives back to our w variable for optimization. We perform stochastic gradient descent, drawing randomly from a distribution of augmentations on a per-batch basis, in order to minimize the expected loss across a distribution of augmentations.

2.4 Image sizes

For some networks, images must be resized or cropped before they can be input. Furthermore following our augmentations, sizes may also have changed. We place a differentiable bilinear resize between the final augmented perturbed image and the input to each model in the ensemble. In this way crops of varying sizes can be taken, blurring ”valid” convolutions without padding can be applied, and models with disparate input sizes can all be optimized simultaneously.

2.5 Examples

Recall that our attacks seek to maximize fooling while adhering to an infinity norm constraint rather than trying to minimize the change to the image. Because of that, the examples in this section are not your typical examples for an adversarial images paper where attacked images are practically indiscernible from the original.

Not only is it very visible that the image has been modified, the attack quite evidently contains a lot of structure.

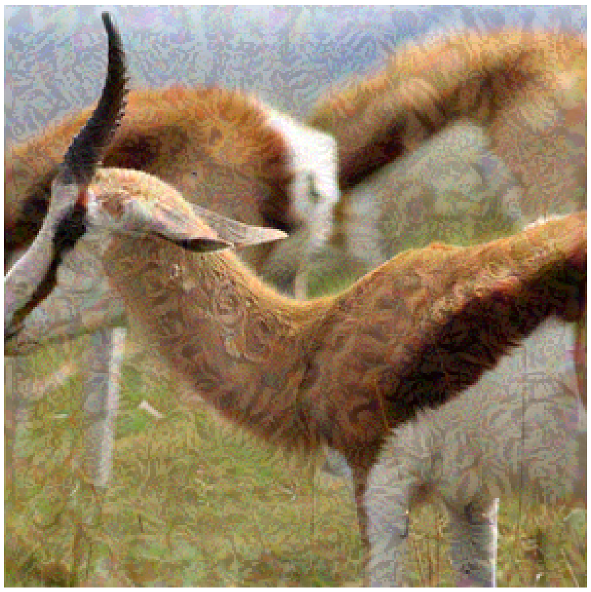


Figure 1: Gazelle from competition dataset, target collie. Optimized at $\epsilon = 16$ for 100 iterations against an ensemble of: adv inception v3 [1], resnet18 [10], and squeezenet1.1 [11]

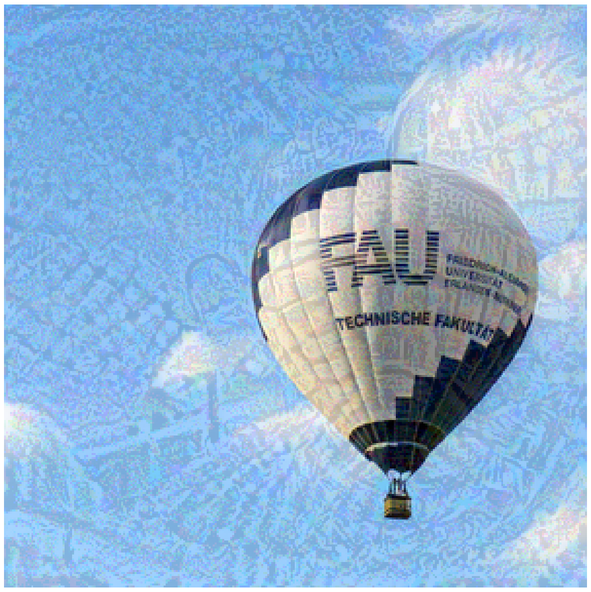


Figure 2: Bolloon from competition dataset, target earthstar. Optimized at $\epsilon = 16$ for 100 iterations against an ensemble of: adv inception v3, resnet18, and squeezenet1.1

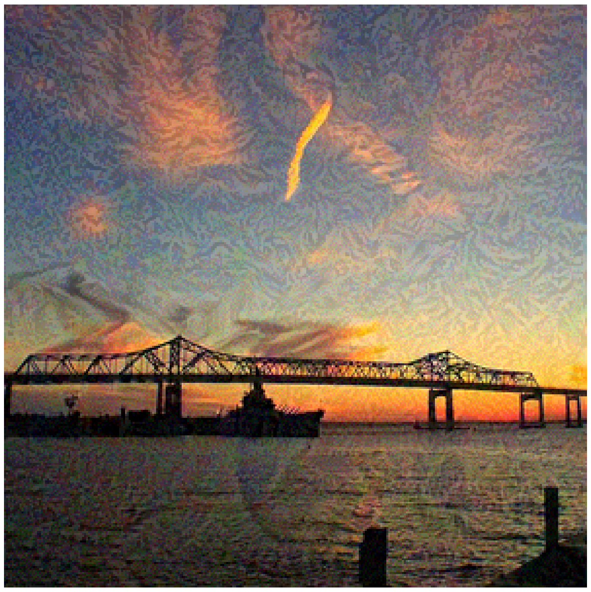


Figure 3: Pier from competition dataset, target pillow. Optimized at $\epsilon = 16$ for 100 iterations against an ensemble of: adv inception v3, resnet18, and squeezenet1.1

3 Our universal attack

Using a variant of our iterative attack, we craft targeted universal perturbations. Single deltas that can be added to any image in order to fool a black box model. We again attack a substitute defense in a white box manner. We maximize transferability by minimizing the loss, using augmentations, and attacking a large, diverse ensemble as the target model.

We draw random batches of images from the ImageNet training set and perform stochastic gradient descent in order to minimize our expected loss for the target class when applying the perturbation to images from the ImageNet distribution.

At attack time, the $tanh(w) \in [-1, 1]$ is multiplied by ϵ, added to the image, and clipped to $[0, 1]$

3.1 Box constraints

In this situation it is impossible to perform the change of variable to satisfy the constraint that the perturbed image remain valid because each image in each batch gives rise to a different constraint. Instead we simply clip: $x + \delta \in [0, 1]$ with a change of variable:

$$\delta_i\ (w_i) = \epsilon\ \cdot tanh(w_i)$$

$$w_i \in [-\infty, \infty]$$

3.2 Examples

In order to visualize the attacks, we take the $tanh(w) \in [-1, 1]$ and scale it to $[0, 255]$ for pixel values. Thus the way to interpret these images is that black regions are subtracting from all pixel values, white regions adding, and green regions are subtracting from RB and adding to G. A mid grey pixel of $(127, 127, 127)$ would represent no change.

All below are trained on batches of 4 from the imagenet training set.

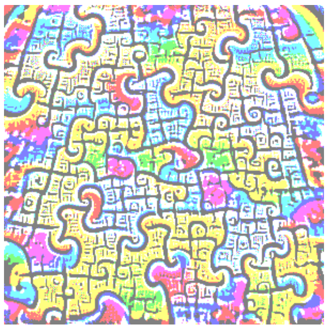


Figure 4: Jigsaw puzzle crafted against an ensemble of: ens adv inception resnet v2 [9], inception v3, adv inception v3, resnet101, dual path net 68 [12], densenet 161 [13], and alexnet. Trained on at $\epsilon = 16$ for 44,600 iterations.

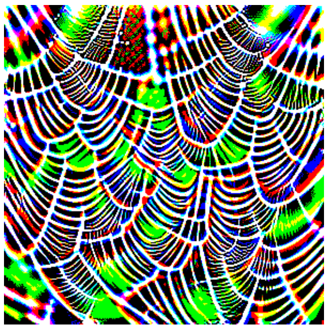


Figure 5: Spider web crafted against an ensemble of: ens adv inception resnet v2, inception v3, adv inception v3, resnet101, dual path net 68b extra, and densenet 161. Trained on at $\epsilon = 8$ for 22,300 iterations.

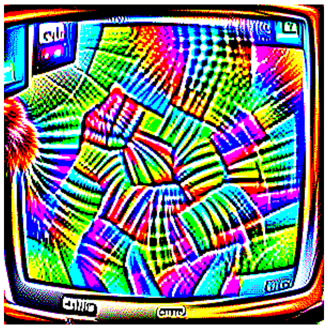


Figure 6: Television crafted against an ensemble of: ens adv inception resnet v2, inception v3, adv inception v3, resnet101, dual path net 68, densenet 161, and alexnet. Trained on at $\epsilon = 16$ for 41,100 iterations.

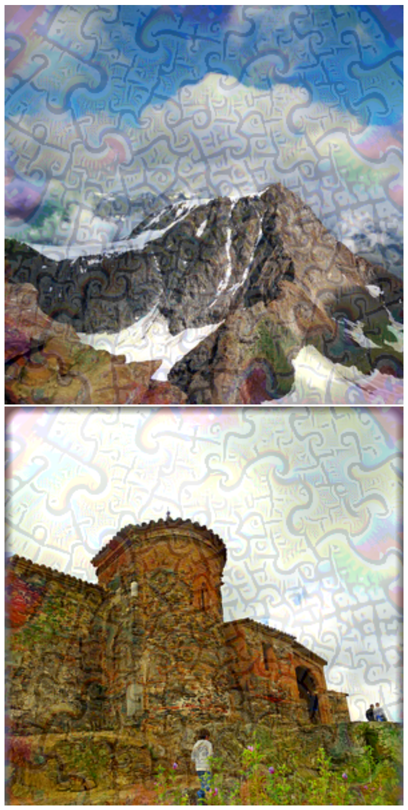


Figure 7: Images attacked by a jigsaw perturbation at $\epsilon = 16$.

3.3 Selective universal attack

Given a set of images, a target model, and a set of universal perturbations, we perform a selective attack. We try each universal perturbation against each image and keep the one that maximized the cross-entropy loss for the target model and the original most likely class for the unperturbed image. During this, we also keep track of whether or not the target model was successfully fooled.

All images for which the target model was not successfully fooled pass to a second stage, where we use our normal iterative attack.

4 Implementation

We perform our attacks using Pytorch. Pytorch is fast and brilliantly easy to work with in a research situation.

We find it best to tune our attacks differently for different ϵ constraints.

We ensemble target models by taking a weighted arithmetic mean of their output log probabilities.

For compactness, any missing details below can be assumed to be the same as stated most recently above.

We did not use augmentations when performing our attacks because the time penalty was too costly, but we made heavy use of them when training universal perturbations.

4.1 Targeted attack

At $\epsilon = 4$ We attack an ensemble of the "ens adv inception resnet v2" from the cleverhans examples, trained as per [9] and the basic inception v3 available from tensorflow-slim, the same that is also included in the cleverhans examples. We weigh the first network at 4 and the later at 1 when ensembling. We weigh the first network more heavily because inception v3 is laughably easy to fool, and we can increase our chances of fooling the first network without compromising. We use the Adam optimizer with a learning rate of 0.2 and betas 0.9 and 0.999. We run a batch size of 20 and typically get in about 35 iterations before the time budget for the batch runs out.

At $\epsilon = 8$ we are able to introduce a third model to our ensemble, the adversarially trained inception v3 from the cleverhans examples. In this case our weights are 1, 2, and 1 for ens adv inception resnet v2, inception v3, and adv inception v3 respectively. These weights allow us to get benefits of having the other networks in the ensemble without sacrificing performance on inception v3. We typically get in about 27 iterations before time runs out.

At $\epsilon = 12$ and 16 we are able to change the weighting to 4, 1, 1, as with a high epsilon we can put greater emphasis on attacking ens adv inception resnet v2 while still maintaining success against inception v3.

4.2 Universal perturbation training

We select target classes by hand. The important observation is that the universal attacks are *semantic* in nature. That is, they are shaped very much by the target class in question, rather than the intricacies of the weights and blind spots of the target models. We therefore seek to select target classes that would make good semantic attacks when combined with an image. Three such classes are:

- Jigsaw puzzle: A jigsaw puzzle can be any image as long it has the characteristic edges
- Spider web: A spider web can hang transparently in front of what would otherwise be an image of something else

- Monitor: A monitor could have anything on it, as long as it has a bezel

We trained against enormous, diverse ensembles including the adversarially trained models made available for the competition, and other base models. One example is en ensemble of ens adv inception resnet v2, inception v3, adv inception v3, resnet101, dual path net 68, densenet 161, and alexnet. Batch sizes were therefore small, typically only 4. We trained at varying ϵ values and for varying iterations. This was a time consuming, ad hoc affair and was not explored scientifically.

4.3 Non-targeted attack

At $\epsilon = 4$ our universal perturbations are too weak to be effective, so we elect to simply use the iterative attack and target the 3rd highest predicted class. The ensemble is the same three models from above, but equally weighted. We get approximately 27 iterations in. We include all three models here, where we did not for the targeted attack because non-targeted attacks are easier to transfer and they transfer better when attacking a larger ensemble.

At $\epsilon = 8$, 12, and 16 we perform the same selective universal attack, falling back to our iterative attack. We select from a dozen of the best universal perturbations we created, targeting brain coral, jigsaw, monitor, and spider web. We evaluate these examples against the same three models, equally weighted. We fall back to an iterative attack, and depending on the situation can get in over 40 iterations per batch.

4.4 Competitive factors

The dynamics of the competition were highly skewed by the sample models and sample code. As was possible to determine by studying the development round results, a significant proportion of the defenses were indistinguishable from the inception v3 sample submission. Furthermore a lot of competitors appeared to be using the ens adv inception resnet v2 out of the box.

We essentially have the opportunity to perform a white box attack on a significant proportion of defenses. Unfortunately our experiments showed that methods for increasing the transferability of our attack cost us more points than we gained because it was simply too easy to perform a white box attack, and too hard to craft a strong black box attack in the time allotted. This meant that we did not choose to use augmentation at evaluation time, and we did not choose to include other models in the ensemble. Both of which increase transferability, but sacrifice performance against inception v3 et al.

The goal, of course, was not to beat the strongest defense, but to beat the most defenses.

References

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna *Rethinking the inception architecture for computer vision* In CVPR, 2016

[2] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. *Inception-v4, inception-resnet and the impact of residual connections on learning* In AAAI, 2017

[3] F. Tramer, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel *Ensemble adversarial training: Attacks and defenses* arXiv preprint arXiv:1705.07204, 2017

[4] Alexey Kurakin, Ian Goodfellow, Samy Bengio *Adversarial Machine Learning at Scale* arXiv:1611.01236

[5] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy *Explaining and Harnessing Adversarial Examples* arXiv:1412.6572

[6] Alexey Kurakin, Ian Goodfellow, Samy Bengio *Adversarial Machine Learning at Scale* arXiv:1611.01236

[7] Nicholas Carlini, David Wagner *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods* arXiv:1705.07263

[8] Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, Qi Zhao *Foveation-based Mechanisms Alleviate Adversarial Examples.* arXiv:1511.06292

[9] Florian Tramr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, Patrick McDaniel *Ensemble Adversarial Training: Attacks and Defenses.* arXiv:1705.07204

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition* arXiv:1512.03385

[11] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size* arXiv:1602.07360

[12] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, Jiashi Feng *Dual Path Networks* arXiv:1707.01629

[13] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten *Densely Connected Convolutional Networks* arXiv:1608.06993

Appendices

A Universal perturbation experiments

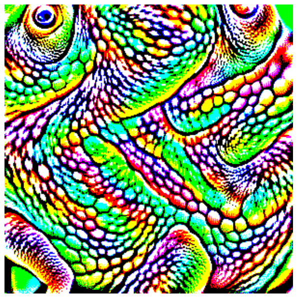
For all experiments we use the 1,000 images from the competition as our testing set. These are 299x299x3 images sourced from Flickr. The images cover 430 unique classes, with no class appearing more than 5 times.

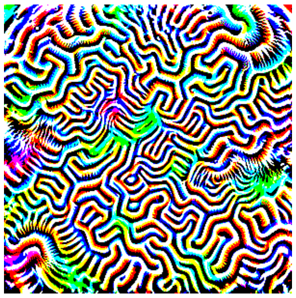
We standardize to the 1,000 imagenet classes by dropping the logits from the 0th ”background” class in the inception models.

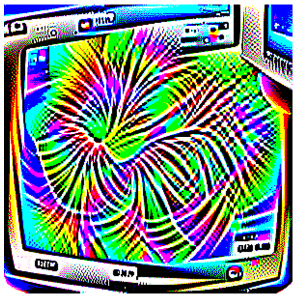
We use:

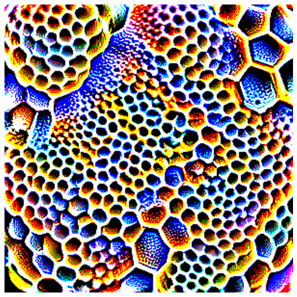
- Inception v3: Ported from tensorflow-slim rather than from the torchvision models.
- Adv Inception v3: Ported from the cleverhans NIPS examples
- Inception Resnet v2: Ported from tensorflow-slim
- Ens Adv Inception Resnet v2: Ported from the cleverhans NIPS examples
- Renset152: From the torchvision models

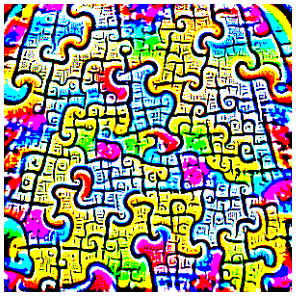
We train universal perturbations with varying ensembles, as per the main paper. Inception v3, Adv Inception v3, and Ens Adv Inception Resnet v2 are always in the ensemble. Inception Resnet v2 and Resnet152 are never in the ensemble, and are included to demonstrate transfer to naturally trained models and models with different architectures. Note that Resnet101 is almost always in the ensemble.

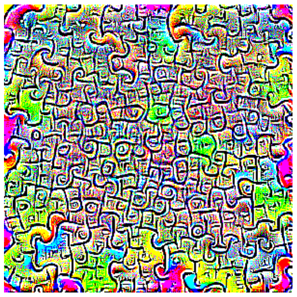

(a) african chameleon

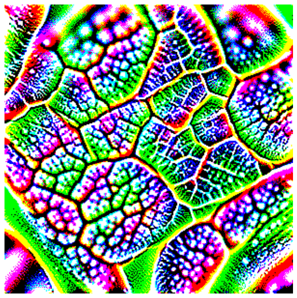

(b) brain coral

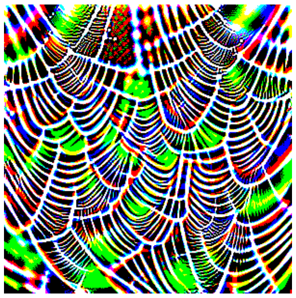

(c) crt


(d) honeycomb


(e) jigsaw1


(f) jigsaw2


(g) ladybug


(h) spider web

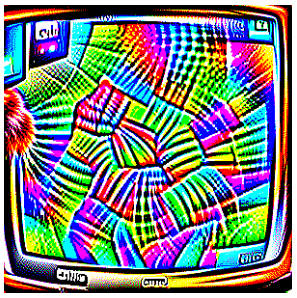

(i) tv

Figure 8: Universal perturbations for experiments

	Inception v3				Inception Resnet v2			
	ϵ				ϵ			
Perturbation	4	8	12	16	4	8	12	16
Natural Images	949				998			
african chameleon	857	576	307	147	928	640	362	196
brain coral	862	576	315	148	946	682	413	210
crt	795	522	289	144	867	591	368	233
honeycomb2	813	529	300	153	900	618	359	198
jigsaw3	876	663	426	254	956	787	581	370
jigsaw5	799	510	285	151	909	627	365	187
ladybug	794	505	289	145	911	622	356	182
spider web	726	407	185	88	936	604	300	135
tv	665	338	145	83	867	506	241	115

	Adv Inception v3				Ens Adv Inc Res v2			
	ϵ				ϵ			
Perturbation	4	8	12	16	4	8	12	16
Natural Images	951				976			
african chameleon	920	776	573	370	974	896	727	485
brain coral	917	758	549	355	969	863	661	411
crt	902	701	539	384	970	875	735	565
honeycomb2	889	699	483	300	971	877	690	476
jigsaw3	927	796	632	470	975	910	768	597
jigsaw5	915	728	529	355	967	836	632	415
ladybug	924	750	550	372	966	841	648	427
spider web	934	746	482	284	964	778	475	235
tv	932	733	478	287	967	716	417	211

	Resnet 152			
	ϵ			
Perturbation	4	8	12	16
Natural Images	945			
african chameleon	888	656	376	185
brain coral	896	665	412	227
crt	899	732	491	316
honeycomb2	853	578	315	159
jigsaw3	833	523	241	98
jigsaw5	771	352	134	62
ladybug	910	775	569	352
spider web	852	576	333	158
tv	897	737	528	334

Figure 9: Top 1 correct for models against universal perturbations at various epsilons. Out of 1,000 images in the competition dev set