Methods for Detecting Adversarial Images and a New Saliency Map

Dan Hendrycks*
University of Chicago
dan@ttic.edu

Kevin Gimpel
Toyota Technological Institute at Chicago
kgimpel@ttic.edu

Abstract

Many machine learning classifiers are vulnerable to adversarial perturbations. An adversarial perturbation modifies an input to change a classifier's prediction without causing the input to appear substantially different to the human perceptual system. We make progress on this AI Safety problem by presenting a three methods to detect adversarial images. Our best detection method reveals that adversarial images place abnormal emphasis on the lower-ranked principal components from PCA. At the end we introduce a new saliency map to increase the transparency of convolutional neural network classification decisions.

1 Introduction

Images can undergo slight yet pathological modifications causing machine learning systems to misclassify, all while humans barely can notice these perturbations. These types of manipulated images are adversarial images (Goodfellow et al., 2015), and their existence demonstrates frailties in machine learning classifiers and a disconnect between human and computer vision.

This unexpected divide can allow attackers complete leverage over some deep learning systems. For example, adversarial images could cause a deep learning classifier to mistake handwritten digits, thereby fooling the classifier to misread the amount on a check (Papernot et al., 2016b; Kurakin et al., 2016). Other fooling data could evade malware detectors or spam filters that use a deep learning backend (Grosse et al.). Worse, generating adversarial images requires no exact knowledge of the deep learning system in use, allowing attackers to achieve consistent control over various classification systems (Szegedy et al., 2014; Papernot et al., 2016a).

A consistently misclassified adversarial image is easy to generate. Say we want an image of a civilian x_{civilian} to be misclassified as a soldier. Then given a neural network model $\mathcal{M}$ and a loss function $L : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$, we can generate an image $x_{\text{adversarial}}$ by minimizing $L_{\mathcal{M}}(x_{\text{adversarial}}, y_{\text{soldier}})+\lambda\|x_{\text{civilian}} - x_{\text{adversarial}}\|_2^2$ using gradient descent to modify the image. We can stop performing gradient descent when $P(y_{\text{soldier}}|x_{\text{adversarial}}, \mathcal{M})$ exceeds a specified threshold (Goodfellow et al., 2015). The result of this procedure is an adversarial image which can be near indistinguishable to the adversarial image, yet it can reliably deceive other deep learning models.

In this paper, we make progress on the AI Safety (Olah et al., 2016) subproblem of detecting adversarial images, and we contribute a new saliency map to make network classification decisions

*Work done while the author was at TTIC. Code available at github.com/hendrycks/fooling more understandable. We start our adversarial images discussion by laying preliminary groundwork before explaining our detection methods. In this preliminary section, we argue that adversarial image generation should be box-constrained and that many previous works may have incorrectly rendered adversarial images. Then we present our first detection method which reveals that adversarial images place abnormally strong emphasis on principal components which account for little variance in the data. Our second methods uses the observation from Hendrycks & Gimpel (2016a), which states that correctly classified and out-of-distribution examples tend to have different softmax distributions, but now we apply this to adversarial images. For our third detection method, we show that adversarial image reconstructions can be worse than clean image reconstructions if the decoder uses classification information. Then introduce a new saliency map (Zeiler & Fergus, 2014; Springenberg et al., 2015) which aims to increase the interpretability of convolutional neural network classification decisions since model transparency is another AI Safety subproblem.

2 Correctly Rendering Adversarial Images

In this section, we argue that adversarial image generators should produce images where each pixel is within $[0, 1]$ because, otherwise, the adversarial perturbations can be conspicuous. Currently, very few papers generate images with these box-constraints. Indeed, Warde-Farley & Goodfellow (2016) state that, "... authors frequently omitted this constraint for simplicity, because the perturbations η are typically small and thus do not move the input significantly far outside the original domain."

While we agree that the perturbations do not move the input *significantly far* outside the domain, they still often move the input outside. Doing so can make the adversarial perturbations visible. To show this, we use the Tiny-ImageNet dataset which is a subset of the ILSVRC-2012 dataset but with 200 object classes. To reduce the chance of bugs experiment, we used a pretrained Tiny-ImageNet classifier and a popular, pre-existing adversarial image generator from github (Voss, 2015). We work with the raw images with each pixel in the range $[0, 255]$, and then we subtract the mean. With these images, we perform adversarial image generation by letting the descent step size be 1. We terminate the generation procedure when the softmax prediction probability for a randomly-chosen class exceeds 50%. For more comparisons, we use the fast gradient sign (Goodfellow et al., 2015) method and let the single step size be 10. In Figure 1, we show the clean and adversarial images created with the iterative and fast gradient sign methods.

Also in Figure 1, we render the adversarial image after rescaling or clipping its pixels be in $[0, 255]$ after adding back the mean image. Noticeably, the adversarial perturbations are no longer visible under a clipping or rescaling postprocessing step. This is a problem because it is easy to incorrectly render an adversarial image and have it appear completely innocuous. Indeed, several adversarial image generators on github rescale or clip the image for viewing while the network evaluates the unclipped and unrescaled image. This is a common problem because many implementations utilizing Torch (Collobert et al., 2011) use the `image.save` or `image.display` functions; the `image.save` function automatically has tensor values "clamped between 0 and 1 before being saved to the disk," per the documentation, and the `image.display` automatically rescales the image. This hides visible artifacts of the adversarial perturbation, and this could imply many papers accidentally overstate the extent to which their adversarial images are pathological.

Absent a clipping or rescaling postprocessing step, the images some have visible perturbations. In fact, we labeled 50 fast gradient sign adversarial Tiny-ImageNet test set images (with step size 10), 50 iteratively generated adversarial images (with step size 1), and 100 clean images as either

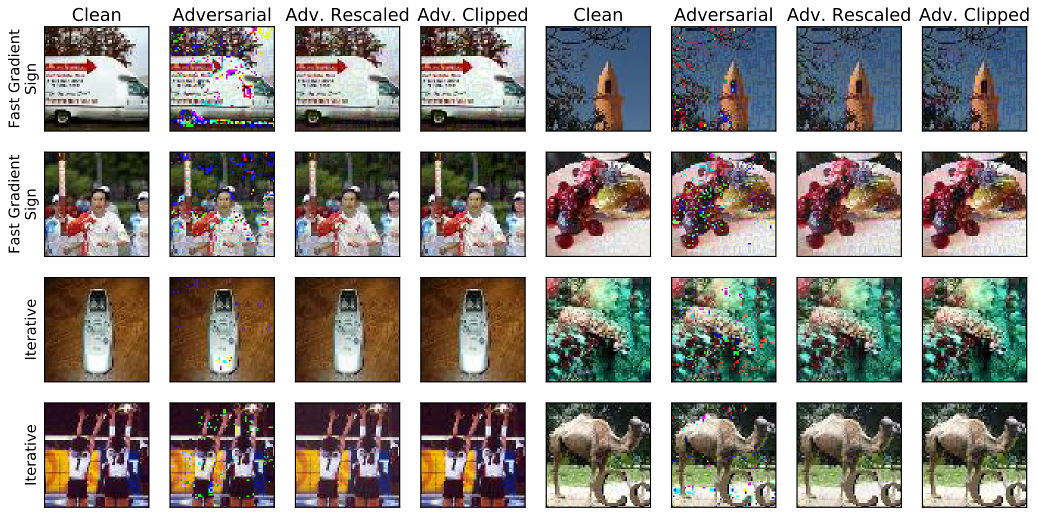

Figure 1: Adversarial examples can have visible perturbations hidden when images are rescaled or clipped after they generated. Examples are randomly chosen.

having "visible adversarial artifacts" or not. Each image was chosen randomly, presented in a random order, and all adversarial images were assumed to have visible artifacts. We correctly labeled the adversarial images with over 90% accuracy and obtained perfect accuracy for the clean images. Furthermore, a previous draft of this paper claimed that adversarial perturbations to YUV CIFAR-10 images can become conspicuous with appreciable probability, but this was mostly because we correctly rendered the adversarial images correctly and did not clip or rescale its values. Consequently, many adversarial perturbations are visible if you do not rescale or clip the images.

A simple solution is to box-constrain the adversarial search process with L-BFGS or clip the adversarial image after each descent step. With these constraints, the adversarial images tend *not* to have visible perturbations when correctly rendered (that is, when they are rendered without rescaling or clipping postprocessing). In view of this, we hope that future research takes care not to rescale or clip adversarial images and, to keep the adversarial images pathological, adversarial image generation should be done with constraints.

3 Detecting Adversarial Images

3.1 Detector Evaluation

To show our three methods for detecting adversarial images, we need to establish our detector evaluation metrics. For detection we have two classes, and the detector outputs a score for both the positive (adversarial image) and negative (clean image) class. To evaluate our detectors, we do not report detection accuracy alone because accuracy is threshold-dependent, and a chosen threshold should vary depending on a practitioner's trade-off between false negatives (fn) and false positives (fp).

Faced with this issue, we employ the Area Under the Receiver Operating Characteristic curve (AUROC) metric, which is a threshold-independent performance evaluation (Davis & Goadrich, 2006). The ROC curve is a graph showing the true positive rate ($\text{tpr} = \text{tp}/(\text{tp} + \text{fn})$) and the false positive rate ($\text{fpr} = \text{fp}/(\text{fp} + \text{tn})$) against each other. Moreover, the AUROC can be interpreted as the probability that an adversarial example has a greater detector score/value than a clean example (Fawcett, 2005). Consequently, a random positive example detector corresponds to a 50% AUROC, and a "perfect" classifier corresponds to 100%.[1]

The AUROC sidesteps the issue of selecting a threshold, as does the Area Under the Precision-Recall curve (AUPR) which is sometimes deemed more informative (Manning & Schütze, 1999). This is because the AUROC is not ideal when the positive class and negative class have greatly differing base rates, which can happen in practice. For this reason, the AUPR is our second evaluation metric. The PR curve plots the precision ($\text{tp}/(\text{tp}+\text{fp})$) and recall ($\text{tp}/(\text{tp}+\text{fn})$) against each other. The baseline detector has an AUPR equal to the precision (Saito & Rehmsmeier, 2015), and a "perfect" classifier has an AUPR of 100%. To show how a classifier behaves with respect to precision and recall, we also show the AUPR.

3.2 PCA Whitening Adversarial Images

We now show the first of three adversarial image detection methods. For the first detector, we need to PCA whiten or "sphere" an input. To do this, we must center the training data about zero, compute the covariance matrix C of the centered data, and find the SVD of C which is $C = U\Sigma V^{\mathsf{T}}$. We can perform PCA whitening by taking an input example x and computing $\Sigma^{-1/2}U^{\mathsf{T}}x$, giving us the PCA whitened input. This whitened vector has as its first entry a coefficient for an eigenvector/principal component of C with the largest eigenvalue. Later entries are coefficients for eigenvectors with smaller eigenvalues. Adversarial images have different coefficients for low-ranked principle components than do clean images.

Adversarial image coefficients for later principle components have consistently greater variance as shown in Figure 3. We use coefficient variance as our sole feature for detecting adversarial images. Specifically, for the Tiny-ImageNet test set, we create fast gradient sign (step size is 10) and iteratively generated (step size is 1) adversarial images. For CIFAR-10 and MNIST test set images, the fast gradient sign adversarial images have a step size of 10/255 since the data is at a different scale, and the iteratively generated images use a step size of 1/255, all while having an ℓ_2 regularization strength of $\lambda = 10^{-3}$. All adversarial examples are clipped after each descent step, following our discussion in the previous section. Finally, we keep the adversarial image only if the softmax prediction probability is above 50% for its randomly chosen target class. Now, we compute the detector "scores" or values by computing the variance of the coefficients for clean and adversarial

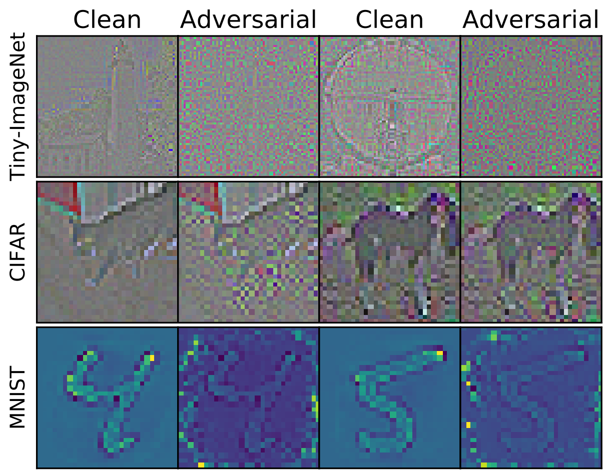

Figure 2: ZCA whitened ($U\Sigma^{-1/2}U^{\mathsf{T}}x$) adversarial images are often visibly distinct from whitened clean images.

[1]A debatable, imprecise interpretation of AUROC values may be as follows: 90%—100%: Excellent, 80%—90%: Good, 70%—80%: Fair, 60%—70%: Poor, 50%—60%: Fail.

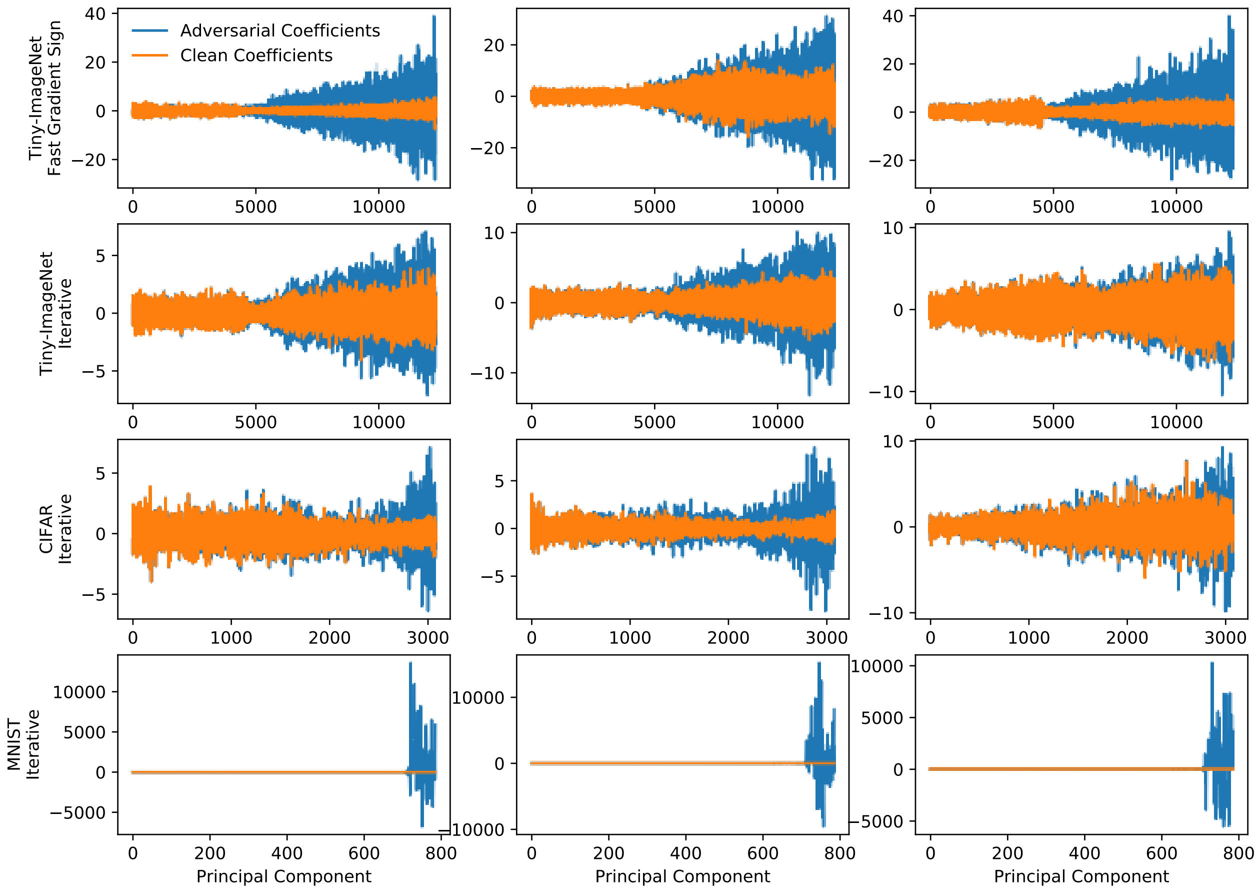

Figure 3: Adversarial images abnormally emphasize coefficients for low-ranked principal components. For example, the 100th "Principle Component" Coefficient is the 100th entry of $\Sigma^{-1/2}U^{\top}x$ for an input image x. Fast gradient sign examples for CIFAR-10 and MNIST are even more abnormal and not depicted due to space. Examples are randomly chosen.

images. These variance values are from a subsection of each whitened input. Specifically, we select the 10000th to final entry of a PCA whitened Tiny-ImageNet input and compute one variance value from this vector subsection. For each CIFAR-10 whitened image, we compute the variance of entries starting at the 2500th entry. For MNIST, we compute the variance of the entries starting at the 700th position. The variances are frequently larger for adversarial images generated with fast gradient sign and iterative methods, as demonstrated in Table 1.

We can also try making adversarial images have typical variances, but this fails. How best to bound the adversarial image search procedure using the fast gradient sign method is not clear. Since each image is generated after one large step, the image's coefficient variance explodes with the one large modification. To show this consider the aforementioned coefficient subsections for whitened Tiny-ImageNet, CIFAR-10, and MNIST examples. All fast gradient sign adversarial images have their coefficient variance beyond *ten* standard deviations from the mean coefficient variance for clean images. Remarkably in the case MNIST, 100% of fast gradient sign images have their coefficient's variance beyond 10 *billion* standard deviations from the mean coefficient

Dataset	Fast Gradient Sign AUROC	Fast Gradient Sign AUPR	Iterative AUROC	Iterative AUPR
Tiny-ImageNet	100.0	100.0	92.4	93.5
CIFAR-10	100.0	100.0	92.8	91.2
MNIST	100.0	100.0	100.0	100.0

Table 1: Variance of entries for whitened adversarial images and whitened clean images differ enough to allow reliable detection. Baseline values are 50%, and "perfect" detectors correspond to 100%. All entries are percentages.

variance of clean images. Similarly, iteratively generated adversarial images fail to obtain typical variances. For MNIST, we repeat the iterative generation procedure described earlier but we bound the search procedure. To show how we bound the search, we establish notation. Let $v_{\text{example}} = \text{Var}((\Sigma^{-1/2}U^\mathsf{T}x_{\text{example}})_{700:784})$, where $700 : 784$ is an array subsection from the 700th to final entry. The clean examples $\text{clean1}, \text{clean2}, \ldots$ have each have the variances $v_{\text{clean1}}, v_{\text{clean2}}$. These variances have the group standard deviation $\sigma_{\text{clean}} = \text{std}([v_{\text{clean1}}, v_{\text{clean2}}, \ldots])$ and mean $\mu_{\text{clean}} = \text{mean}([v_{\text{clean1}}, \ldots])$. We desire that our adversarial example has its coefficient variance within one standard deviation σ_{clean} of the mean coefficient variance for clean examples μ_{clean}. This requirement is established by adding the logarithmic barriers $-\log(\mu_{\text{clean}} + \sigma_{\text{clean}} - v_{\text{adversarial}})$ and $-\log(v_{\text{adversarial}} - \mu_{\text{clean}} + \sigma_{\text{clean}})$ to the loss. Under these constraints, the search procedure could not generate any adversarial MNIST digit. We did not try converting these constraints for Tiny-ImageNet because the popular codebase taken from github is substantially different from mainstream libraries. However, if we convert these constraints for CIFAR-10 images, then 92% of the randomly chosen clean images could not be converted to an adversarial image with satisfactory coefficient variance. In all, encouraging typical coefficient variances appears difficult for fast gradient sign and iterative adversarial images.

3.3 Softmax Distributions of Adversarial Images

Hendrycks & Gimpel (2016a) show that many out-of-distribution examples tend to have different softmax output vectors than in-distribution examples. For example, in-distribution examples tend to have a greater maximum probability than softmax vectors for out-of-distribution examples. Similarly, the KL divergence of the softmax distribution p from the uniform distribution u is larger for clean examples than for out-of-distribution examples. Therefore, adversarial examples which stop being modified when the predicted class probability is around 50% are often easily distinguishable from clean examples since the maximum softmax probability for clean examples tend to be larger than 50% across many tasks. We can easily detect whether an image is adversarial by considering the resulting softmax distribution for adversarial image generation algorithms.

Thus in order to make an adversarial image which has a softmax distribution similar to those of clean examples, we constrain the adversarial image generation procedure. In this experiment, we take correctly classified CIFAR-10 test images and calculate the standard deviation of the KL divergence of the softmax output distribution p from the uniform distribution over 10 elements ($\sigma_{\text{KL}[p\|u]}$). We create an adversarial image iteratively, taking a step size of 1/255 with a regularization strength of 10^{-3} and clipping the image after each step so there is no disconnect between the adversarial image and how it is rendered. We then bound the search procedure with a radius

Distance from mean KL-Div	ℓ_1	ℓ_2	ℓ_∞	Creation Success
$r = \infty$	23.4	0.43	0.081	100%
$r = \sigma_{\text{KL}[p\|u]}$	32.3	1.38	0.103	83%
$r = \frac{1}{2}\sigma_{\text{KL}[p\|u]}$	30.9	1.08	0.101	63%
$r = \frac{1}{4}\sigma_{\text{KL}[p\|u]}$	44.2	2.82	0.153	70%

Table 2: Constraints degrade the pathology of fooling images. Value σ is a standard deviation.

r proportional to $\sigma_{\text{KL}[p\|u]}$ about the mean KL divergence of p from u ($\mu_{\text{KL}[p\|u]}$). We bound the search procedure by adding logarithmic barriers to the loss. With these bounds, the generator must construct an image with a prediction probability greater than 50% within 1000 steps and with a typical KL divergence. If it succeeds, we have a "Creation Success." In Table 2, we observe that constraining the search procedure with finite radii greatly increases the ℓ_2 distance. Therefore, to get a typical softmax distribution, the adversarial image quality must decline.

3.4 Reconstructing Adversarial Images with Logits

Our final detection method comes from comparing inputs and their reconstructions. For this experiment, we train an MNIST classifier and attach an auxiliary decoder to reconstruct the input. Auxiliary decoders are sometimes known to increase classification performance (Zhang et al., 2016). We use a hidden layer size of 256, the Adam Optimizer (Kingma & Ba, 2015) with default parameters, and the GELU nonlinearity (Hendrycks & Gimpel, 2016b). The autoencoder bottleneck has only 10 neurons. Crucially, we also feed the logits, viz. the input to a softmax, into the bottleneck layer. After training, we create adversarial images by an iterative procedure like before, except that we increase the ℓ_2 regularization strength 1000-fold to $\lambda = 1$. Now, as evident in Figure 4, the reconstructions for adversarial images look atypical. From this observation, we build our final detector. The detector's input score is mean difference of the input image and its reconstruction. This difference is greater between adversarial images and their reconstruction than for clean examples, so we can detect an adversarial example from a clean example with an AUROC of 96.2% and an AUPR of 96.6% where the baseline values are 50%. Thus the differences between inputs and their reconstructions can allow for adversarial image detection.

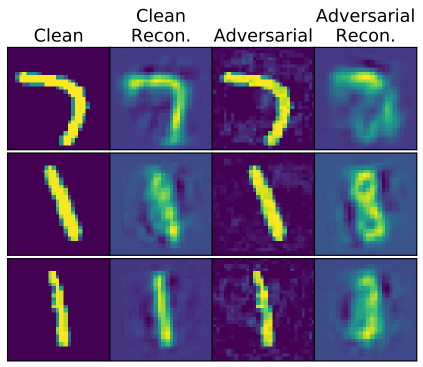

Figure 4: Adversarial image reconstructions are of lower quality than clean image reconstructions.

4 A Saliency Map

Let us now stop considering adversarial images and turn to a different AI Safety goal. This goal is to make neural networks more interpretable. A common way to understand network classifications

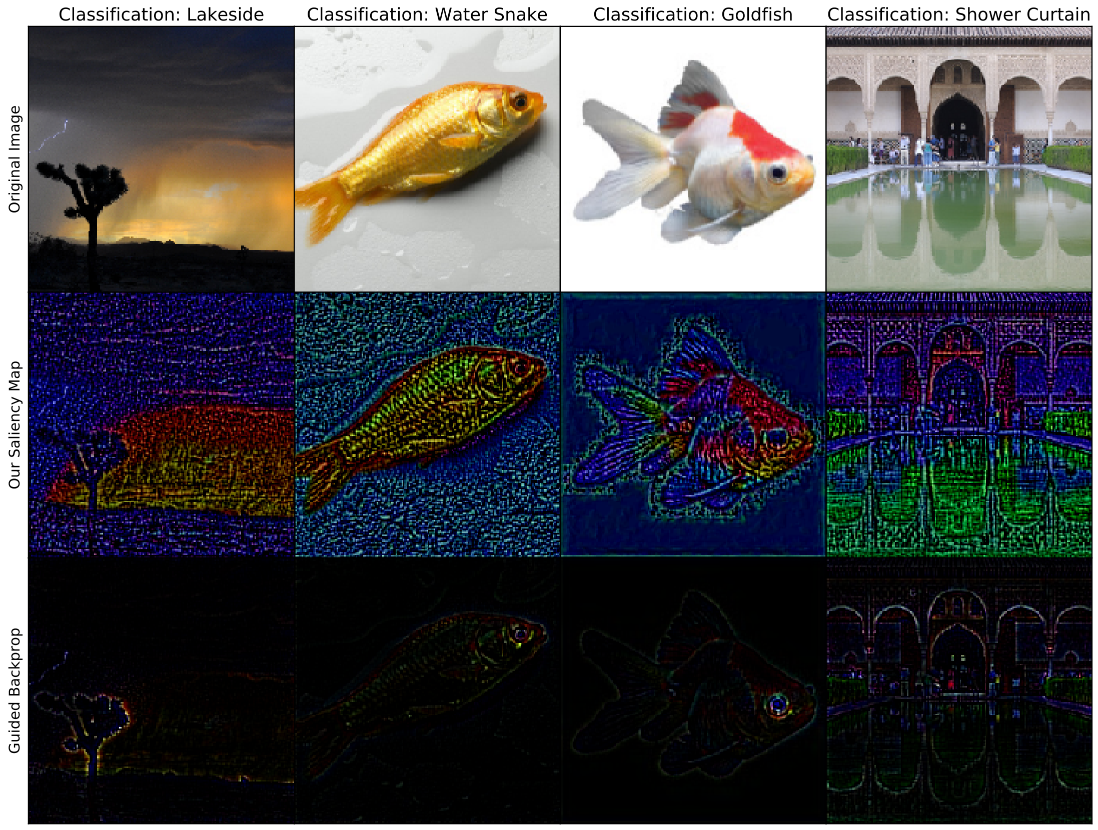

Figure 5: Our saliency map reveals more saliency details than guided backprogagation.

is through saliency maps. And a simple way to make a saliency map is by computing the gradient of the network's logits with respect to the input and backpropagating the error signal without modifying the weights. Then after the error signal traversed backward through the network, we can display the resulting gradient as an image, and this shows us salient parts of the image. A recently proposed technique to improve saliency maps is guided backpropagation Springenberg et al. (2015). To understand guided backpropagation, let us establish some notation. Let $f_i^{l+1} = \text{ReLU}(x_i) = \max(0, x_i)$ and $R_i^{l+1} = \partial f^{\text{output}} / \partial f_i^{l+1}$. Now while normally in training we let $R_i^l = (f_i^l > 0) R_i^{l+1}$, in guided backpropagation we let $R_i^l = (f_i^l > 0)(R_i^{l+1} > 0) R_i^{l+1}$. We can improve the resulting saliency map significantly if we instead let $R_i^l = (f_i^l > 0)(R_i^{l+1} > 0)$, leading to our saliency map.[2]

We demonstrate the results in Figure 5 using a pretrained VGG-16 model (Dieleman et al., 2015; Simonyan & Zisserman, 2015). The positive saliency map consists of the positive gradient values. In the first column, the network classifies this desert scene as a lakeside. Our saliency map reveals that coloring the clouds bluer would increase the logits, as would making the sunlit sky more orange. From this we can surmise that the clouds serve as the lakeside water, and the sky the shore. Next, the fish construed as a water snake has a saliency map where the background resembles water, and the fish's scales are greener and more articulated. Also with our saliency map, we can see which white regions of the fish become orange and would most increase the logits.

[2]Note that this technique works for other nonlinearities as well. For example, if we use a Gaussian Error Linear Unit and have $g_i^{l+1} = \text{GELU}(x_i)$, then letting $R_i^l = g'(x_i) g'(R_i^{l+1})$ produces similar saliency maps.

Meanwhile, guided backpropagation shows the fish's eye. Last, our saliency map makes it evident that greener water would further increase the logits suggesting its role in the image's shower curtain misclassification. Overall, this new saliency map allows for visualizing clearer sources of saliency.

5 Conclusion

In this work, we showcased techniques for detecting adversarial images and visualizing salient regions of an image. Future work could include more reliable methods for detection. For example, not only was the variance larger for adversarial examples, but the coefficient magnitude continued to grow as the principal components became lower in their order, so a future technique could incorporate magnitude growth information for detection. Overall, even tough adversarial perturbations can be pathological to the human eye, they can greatly affect some statistics of the underlying image enabling their detection.

Acknowledgments

We would like to thank Nicolas Papernot and Chris Olah for their helpful feedback. Thank you to Greg Shakhnarovich for numerous insightful suggestions. We would also like to thank the NVIDIA Corporation for donating GPUs used in this research.

References

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proc. of International Conference on Machine Learning (ICML)*, 2006.

Sander Dieleman, Jan Schlter, Colin Raffel, Eben Olson, Sren Kaae Snderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, and Brian McFee diogo149. Lasagne: First release. 2015.

Tom Fawcett. *An introduction to ROC analysis.* Pattern Recognition Letters, 2005.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv.*

Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks.* arXiv, 2016a.

Dan Hendrycks and Kevin Gimpel. *Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units.* arXiv, 2016b.

Diederik Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* International Conference for Learning Representations (ICLR), 2015.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. *Adversarial Examples in the Physical World.* arXiv, 2016.

Chris Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

Chris Olah, Dario Amodei, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv*, 2016.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. *Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples.* arXiv, 2016a.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swam. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security & Privacy*, 2016b.

Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. In *PLoS ONE*. 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *International Conference on Learning Representations (ICLR)*, 2015.

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

Catalin Voss. Visualizing and breaking convnets. 2015. URL `https://github.com/MyHumbleSelf/cs231n/blob/master/assignment3/q4.ipynb`.

David Warde-Farley and Ian Goodfellow. *Adversarial Perturbations of Deep Neural Networks.* 2016.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014.

Yuting Zhang, Kibok Lee, and Honglak Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. International Conference on Machine Learning (ICML), 2016.