# Visible Progress on 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. We make progress on this AI Safety problem by a simple conversion to the YUV colorspace. After demonstrating that adversarial perturbations which modify YUV images are more conspicuous and less pathological, we introduce a new saliency map to better understand misclassification.

## 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 [3], and their existence demonstrates some fragility 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 [7, 6]. Other adversarial data could evade malware detectors or spam filters that use a deep learning backend. 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 [10].

A consistently misclassified adversarial image is easy to generate. Say we want an image of a civilian $x_{\text{civilian}}$ 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 $x_{\text{fool}}$ by minimizing $L_{\mathcal{M}}(x_{\text{fool}}, y_{\text{soldier}}) + \lambda \|x_{\text{civilian}} - x_{\text{fool}}\|_2^2$ using gradient descent to modify the image. We can stop performing gradient descent when $P(y_{\text{soldier}}|x_{\text{fool}}, \mathcal{M})$ exceeds a specified threshold [3]. The result of this procedure is an adversarial image which is near indistinguishable to the adversarial image, yet it can reliably fool other deep learning models.

In this paper, we make progress on the AI Safety [1] subproblem of adversarial images and we contribute a new saliency map to allow for more interpretability of convolutional neural networks. In this work's first section, we simply convert training data to the YUV color space and train a convolutional neural network classifier on this data. This makes the adversarial image generation processes need to make conspicuous modifications to the image in order to fool the network. That is, converting to YUV makes adversarial perturbations more visible and therefore limits their effectiveness. Building on this technique, we then demonstrate a preprocessing method which makes the network more robust to adversarial images. Next, because visualizing neural networks can provide for increased interpretability, predictability, and safety, we present a new

*Work done while the author was at TTIC. Code available at github.com/hendrycks/fooling

"convolution transpose" or "deconvolution" technique to visualize what modifications to an image would most affect the logits.

## 2 Neural Network Protection with YUV and $\ell_2$ Pooling

We demonstrate that training a neural network on YUV images makes pathological perturbations visible, thereby diminishing the harm of adversarial images. In this experiment we train a 32-layer residual network [4, 2] on images in the YUV colorspace, which is a colorspace that separates chrominance and luminance. In order to fool this network, we use an adversarial image generator which takes a clean image and randomly chooses a target label. By gradient descent, the image generator minimizes the loss and stops immediately when the network's confidence in the image is greater than 50%. The loss has a regularization penalty of $10^{-4}$ (that is, $\lambda = 10^{-4}$ from the introduction), and the fooler descends on this loss with a step size of $5 \times 10^{-3}$. We let the fooler generate adversarial images by modifying the YUV image directly. In Figure 1, we observe that the resulting fooling images have perceptible perturbations, whereas previously they were unnoticeable. Moreover, since converting from RGB to YUV does not destroy the image's information, this method of protection has *no impact on accuracy*. In sum, a neural network trained on YUV images will correspond to "fooling" images which are visibly out-of-sample.

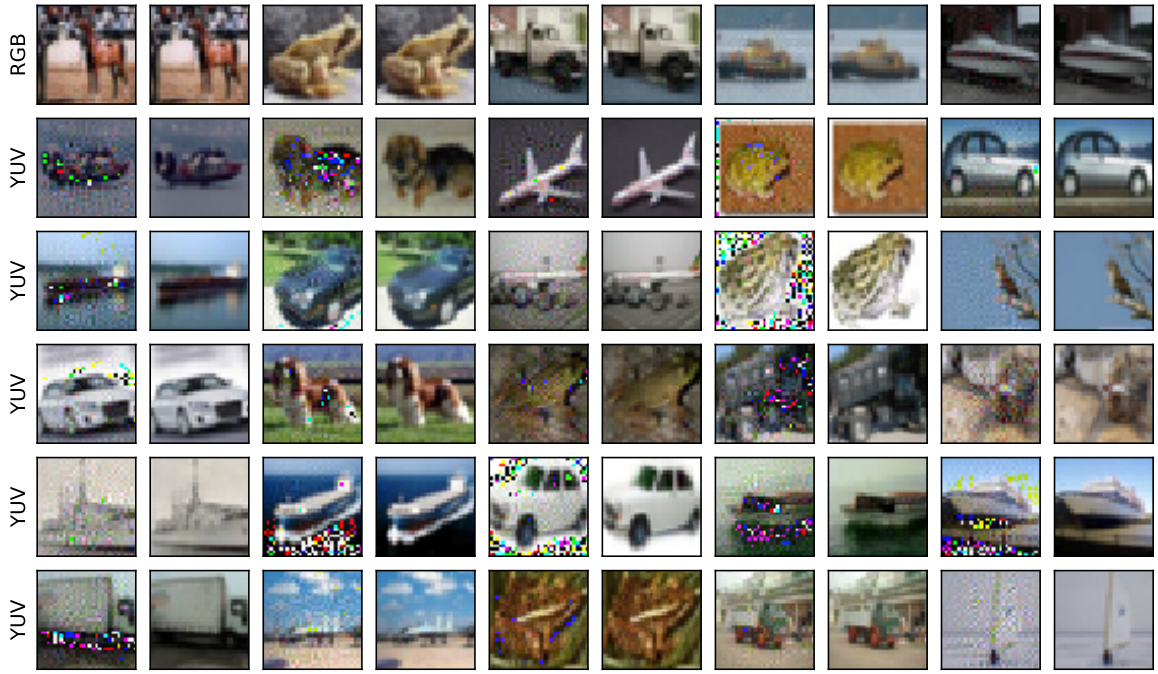


Figure 1: Adversarial Images for a 32-layer Residual Network. In the top row are typical adversarial images for a residual network trained on RGB images, with the clean image right of the fooling image. These are shown for comparison purposes. The rows below are adversarial images for a residual network trained on YUV images, where each fooling image has the clean image at the right. Examples are randomly generated and not cherry-picked.

### 2.1 An Extension

Although the adversarial images now have perceptible perturbations, we can force the adversarial image generator to make even more image alterations in its attempt to fool the network. This is possible by way of a simple preprocessing technique. To motivate our preprocessing technique, recall our example of a model $\mathcal{M}$ being attacked so that it classifies a civilian image as a soldier. For simplicity, assume $\mathcal{M}$ is a logistic regression classifier with weights $w$ and low outputs are classified as soldiers. Following the analysis of [3], we add the noise $\varepsilon$ to $x_{\text{civilian}}$ to get obtain a fooling image $x_{\text{fool}} = x_{\text{civilian}} + \varepsilon$ and decrease $\sigma(w^{\mathsf{T}}(x_{\text{civilian}} + \varepsilon)) = \sigma(w^{\mathsf{T}}x_{\text{civilian}} + w^{\mathsf{T}}\varepsilon)$. Then we need only let $\varepsilon_i = -a \cdot \text{sign} w_i$, $a$ a small positive factor, to increase the probability of classifying a civilian as a soldier. If $w$ is $n$-dimensional with an average entry magnitude of $m$, then the impact of adding $\varepsilon$ on the inner product is $-amn$. Consequently, minuscule perturbations have an outsized impact on the classification for high-dimensional inputs.

However, if the $x_{\text{fool}}$'s entries are in $[0, 1]$, then we could elementwise square the pixels of $x_{\text{fool}}$ without much loss of information. In effect, the noise $\varepsilon$ must then decrease $\sigma(w^{\mathsf{T}}x_{\text{fool}} \odot x_{\text{fool}}) = \sigma(w^{\mathsf{T}}x^2_{\text{civilian}} + 2w^{\mathsf{T}}(x_{\text{civilian}} \odot \varepsilon) + w^{\mathsf{T}}\varepsilon^2)$, $\odot$ the elementwise product, and to accomplish this we need $\varepsilon_i = -a \cdot \text{sign}\,[(x_{\text{civilian}})_i w_i])$. Thus here $\varepsilon$ must take on values which directly dissociates the original image and the weights rather than simply move in a direction opposite the weights. Because we elementwise squared the image, the adversarial noise may be forced to modify the image visibly if it is to fool the network.

Another way to compel the adversarial noise to need to conspicuously alter the image is by slightly blurring the pixels. Doing so requires the adversarial perturbations to coordinate with other adversarial perturbations spatially. Therefore, if we slightly blur the squared pixels and take the square root of the resulting image, we have $\ell_2$ pooled the image without decreasing the dimension. The new pooled image pressures adversarial image generators to make spatially coordinated perturbations.

We apply this idea on a 32-layer Residual Network trained on YUV images where each dimension is rescaled to $[0, 1]$. We square the pixels, blur them them with a Gaussian filter with standard deviation 0.7, and take the square root of this image. We feed this preprocessed image into the residual network trained on clean, rescaled YUV images. Further, we feed in the fooling image without this preprocessing and let the softmax probability for the loss be the average of these two softmaxes. This is the loss the fooler must optimize in this experiment. With this preprocessing, the adversarial images have an average $\ell_2$ distance of 2.44 from the clean image; without this preprocessing, it is 1.75. Therefore simple preprocessing further limits the effectiveness of adversarial image generators.

## 3 A Saliency Map

Another goal of AI Safety is making neural networks more interpretable. A common way to understand convolutional networks and their misclassifications is through saliency maps. 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 the network, we have a visualization of the most salient elements of an image by plotting the gradient as an image. A recently proposed technique to improve saliency maps is guided backpropagation [9]. 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. We demonstrate the results in Figures 2 and 3 using a VGG-16 model trained on ImageNet [2, 8]. The positive saliency map consists of the positive gradient values. Note that this technique works for other nonlinearities

as well. For example, if we use a Gaussian Error Linear Unit [5] 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. Overall, this new saliency map allows for visualizing clearer causes of misclassification.

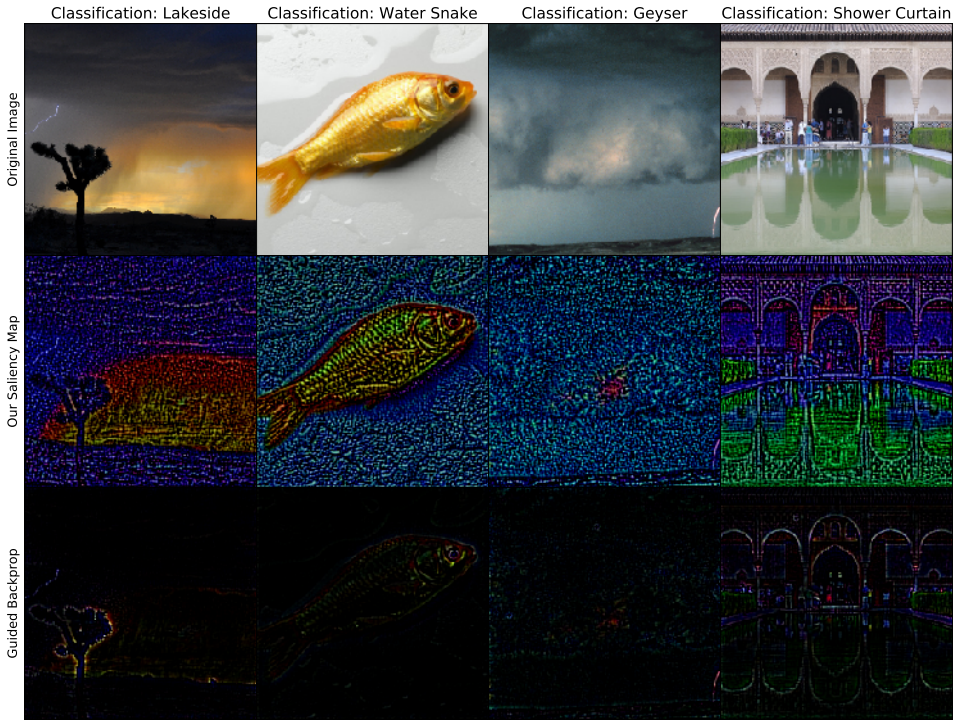


Figure 2: In the first column, the network classifies this desert scene as a lakeside. Our saliency map reveals the lightning may be interpreted as a wave crest, and the sky is the shore. Next, the fish construed as a water snake has a saliency map where the background resembles water, and its the fish's scales are greener and more articulated. The clouds misclassified as a geyser is plausibly because the image center is considered the geyser's opening. Last, our saliency map makes it evident that the green water is leading to the shower curtain misclassification.

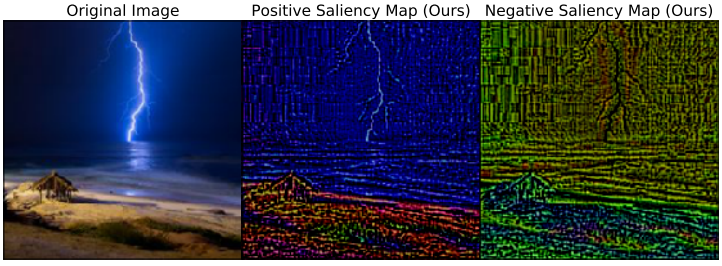


Figure 3: Reasonably classified images can be better understood by the saliency map too. This scene is classified as a lake shore, and we see that making the shore bluer would most decrease the logits. Curiously, the sky texture from this saliency map consists of tiles.

# 4 Conclusion

In this work, we demonstrated that neural networks trained on YUV images correspond to far less pathological adversarial examples. Furthermore, we contributed a new saliency map which allows us to better interpret the behavior of convolutional neural networks. Future work on the adversarial images AI Safety problem could include combining the YUV image conversion presented here with previous defenses against adversarial image attacks.

# Acknowledgments

We would like to thank Greg Shakhnarovich for numerous helpful suggestions. We would also like to thank the NVIDIA Corporation for donating GPUs used in this research.

# References

[1] Chris Olah & Dario Amodei & Jacob Steinhardt & Paul Christiano & John Schulman & Dan Mané. (2016) Concrete Problems in AI Safety. *In arXiv.*

[2] 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 & diogo149 & Brian McFee & Hendrik Weideman & takacsg84 & peterderivaz & Jon instagibbs & Dr. Kashif Rasul & CongLiu & Britefury & Jonas Degrave. (2015) Lasagne: First release.

[3] Ian J. Goodfellow & Jonathon Shlens & Christian Szegedy. (2015) Explaining and Harnessing Adversarial Examples. *In International Conference on Learning Representations (ICLR).*

[4] Kaiming He & Xiangyu Zhang & Shaoqing Ren & Jian Sun. (2015) Deep Residual Learning for Image Recognition. *In Neural Information Processing Systems (NIPS).*

[5] Dan Hendrycks & Kevin Gimpel. (2016) Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *In arXiv.*

[6] Alexey Kurakin & Ian J. Goodfellow & Samy Bengio. (2016) Adversarial Examples in the Physical World. *In arXiv.*

[7] Nicolas Papernot & Patrick McDaniel & Xi Wu & Somesh Jha & Ananthram Swam. (2016) Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *In IEEE Symposium on Security & Privacy.*

[8] Karen Simonyan & Andrew Zisserman. (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. *In International Conference on Learning Representations (ICLR).*

[9] Jost Tobias Springenberg & Alexey Dosovitskiy & Thomas Brox & Martin Riedmiller. (2015) Striving for Simplicity: The All Convolutional Net. *In International Conference on Learning Representations (ICLR).*

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