A. Training Details

Models trained on CIFAR-10. We trained the Wide-ResNet-28-10 model (Zagoruyko & Komodakis, 2016) using standard data augmentation of flips, horizontal shifts and crops in addition to Gaussian noise independently sampled for each image in every minibatch. The models were trained with the open-source code by Cubuk et al. (2018) for 200 epochs, using the same hyperparameters which we summarize here: a weight decay of 5e-4, learning rate of 0.1, batch size of 128. The learning rate was decayed by a factor of 0.2 at epochs 60, 120, 160.

Models trained on ImageNet. The Inception v3 model (Szegedy et al., 2016) was trained with a learning rate of 1.6, batch size of 4096, and weight decay of 8e-5. During training, Gaussian noise was independently sampled for each image in every minibatch. The models were trained for 130 epochs, where the learning rate was decayed by a factor of 0.975 every epoch. Learning rate was linearly increased from 0 to the value of 1.6 over the first 10 epochs.

B. Full Corruption Robustness Results

In this section we examine the corruption robustness of both adversarially trained models and models trained with Gaussian data augmentation. Full results are shown in Tables 1, 2. We highlight several interesting findings from these experiments.

- On CIFAR-10-C, Gaussian data augmentation outperforms adversarial training on the overall benchmark. However, adversarial training is better on all of the blurring corruptions.

- The publicly released ImageNet-C dataset as .jpeg files is significantly harder than the same dataset when the corruptions are applied in memory. It appears that this is due to additional artifacts added to the image from the JPEG compression algorithm (see Figures 7 and 8). Future work should make care of this distinction when comparing the performance of their methods, in particular we note that the results in (Geirhos et al., 2018; Hendrycks & Dietterich, 2019) were both evaluated on the jpeg files.

- Both adversarial training and Gaussian data augmenation significantly degrade performance on the severe fog and constrast corruptions (Tables 3, 4). This highlights the importance of evaluating on a broad suite of corruptions as simply evaluating on worst-case l_p perturbations or random noise will not expose all failings of a model. This also highlights the need for developing methods that improve robustness to all corruptions. Towards this end the exciting new "Stylized ImageNet"(Geirhos et al., 2018) data augmentation process achieves moderate improvements on all corruptions, at least on the publicly released .jpeg files.

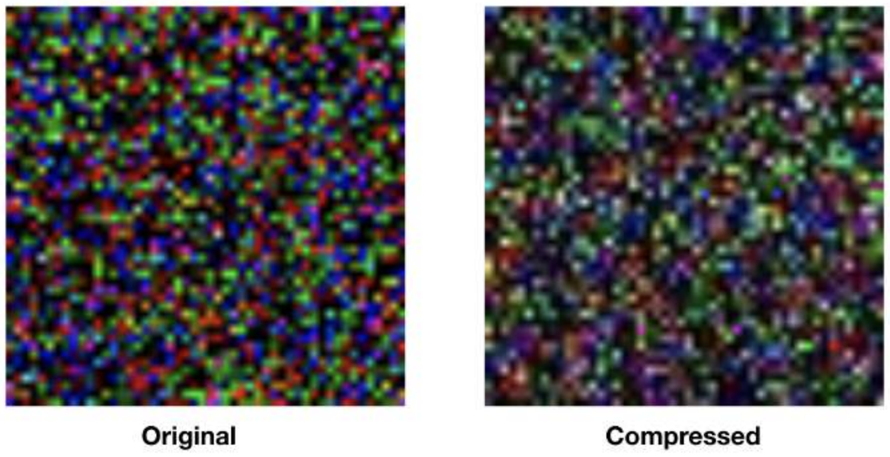

Figure 7. Visualizing the effects of jpeg compression on white noise. The subtle difference between the compressed and uncompressed images is enough to degrade model performance on several of the Imagenet-C corruptions.

Table 1. Measuring the improvements of Gaussian data augmentation on corruption robustness for ImageNet-C. For this table we evaluate both on corruptions in memory to the existing ImageNet validation set using the code at `https://github.com/hendrycks/robustness` and on the compressed version of the dataset from `https://drive.google.com/drive/folders/1HDVw6CmX3HiG0ODFtI75iIfBDxSiSz2K?usp=sharing`. We found that model performance when the corruption was applied in memory is higher than performance on the publicly released .jpeg files that already have the corruptions applied to them. Unfortunately, we were unable to evaluate all corruptions due to issues installing some of the dependencies, these are marked with a ?. All numbers are model accuracies averaged over the 5 corruption severities.

		Noise			Blur			
Training	All	Gaussian	Shot	Impulse	Defocus	Glass	Motion	Zoom
Vanilla InceptionV3	45.0	40.3	38.7	38.0	40.3	26.4	?	31.6
Gaussian ($\sigma = 0.4$)	**52.6**	**67.5**	**67.5**	**66.4**	**43.4**	**39.4**	?	**33.0**
	Weather				Digital			
Training	Snow	Frost	Fog	Brightness	Contrast	Elastic	Pixelate	JPEG
Vanilla InceptionV3	?	?	**60.0**	68.6	**45.2**	46.8	42.8	56.2
Gaussian ($\sigma = 0.4$)	?	?	54.0	**68.8**	39.0	**51.6**	**51.8**	**63.6**
		Noise (Compressed)			Blur (Compressed)			
Training	All	Gaussian	Shot	Impulse	Defocus	Glass	Motion	Zoom
Vanilla InceptionV3	38.8	36.6	34.3	34.7	31.1	19.3	35.3	30.1
Gaussian ($\sigma = 0.4$)	**42.7**	**40.3**	**38.8**	**37.7**	**32.9**	**29.8**	**35.3**	**33.1**
	Weather (Compressed)				Digital (Compressed)			
Training	Snow	Frost	Fog	Brightness	Contrast	Elastic	Pixelate	JPEG
Vanilla InceptionV3	33.1	34.0	**52.4**	66.0	**35.9**	47.8	38.2	50.9
Gaussian ($\sigma = 0.4$)	**36.6**	**43.5**	52.3	**67.1**	35.8	**52.2**	**47.0**	**55.5**

Table 2. Comparing the corruption robustness of adversarial training and Gaussian data augmentation on the CIFAR-10-C dataset. For this table we evaluate on the publicly release .npy files found at `https://drive.google.com/drive/folders/1HDVw6CmX3HiG0ODFtI75iIfBDxSiSz2K?usp=sharing`. Unlike the ImageNet-C dataset which was released as .jpeg files, there was no additional noise applied when saving the images as .npy files. All numbers are model accuracies averaged over the 5 corruption severities.

		Noise			Digital			
Training	All	Speckle	Shot	Impulse	Contrast	Elastic	Pixelate	JPEG
Vanilla Wide-ResNet-28-10	76.3	62.8	59.3	53.3	**92.2**	**84.8**	74.0	77.2
Adversarialy Trained	80.9	81.8	82.8	68.8	77.0	81.8	85.3	85.4
Gaussian ($\sigma = 0.1$)	**81.2**	**91.1**	**91.8**	81.5	58.9	82.2	**89.0**	**90.0**
Gaussian ($\sigma = 0.4$)	74.7	84.6	84.6	**84.5**	41.5	75.4	81.2	82.9
		Weather			Blur			
Training	Snow	Fog	Brightness	Defocus	Glass	Motion	Zoom	Gaussian
Vanilla Wide-ResNet-28-10	83.3	**90.4**	**94.0**	**85.5**	51.1	**81.2**	79.9	75.3
Adversarialy Trained	82.6	72.7	87.1	83.5	**80.2**	80.5	**82.8**	**82.1**
Gaussian ($\sigma = 0.1$)	**87.3**	71.5	91.8	80.0	79.6	71.6	77.2	74.2
Gaussian ($\sigma = 0.4$)	78.0	51.8	80.1	77.0	77.9	72.0	74.8	74.4

Table 3. Detailed results for the fog and contrast corruptions on ImageNet-C highlighting the effect of the severity on both the compressed and uncompressed versions of the data. When the corruption is applied in memory, Gaussian data augmentation degrades performance in comparison to a clean model. However, when evaluating on the compressed version of this dataset this degradation in comparison to the clean model is minimized.

corruption	clean	trained on noise	clean (compressed)	trained on noise (compressed)
contrast-1	**68.198**	66.528	62.502	**63.876**
contrast-2	**63.392**	60.634	55.626	**57.308**
contrast-3	**53.878**	47.57	42.024	**42.434**
contrast-4	**30.698**	17.34	**16.172**	13.122
contrast-5	**9.746**	2.798	**3.362**	2.07
fog-1	**67.274**	65.148	61.334	**62.91**
fog-2	**63.77**	60.398	56.51	**57.746**
fog-3	**59.51**	53.752	51.188	**51.292**
fog-4	**58.098**	51.34	**50.064**	49.324
fog-5	**50.996**	39.586	**42.874**	40.34

Table 4. Detailed results for the fog and contrast corruptions on CIFAR-10-C. Both adversarial training and Gaussian data augmenation significantly degrade performance on these corruptions.

corruption	clean	adv	Gaussian (0.1)	Gaussian (0.4)
contrast-0	**94.73**	86.65	90.51	76.45
contrast-1	**94.22**	84.59	77.12	50.71
contrast-2	**93.67**	82.09	63.71	36.49
contrast-3	**92.51**	76.40	43.74	25.96
contrast-4	**85.66**	55.29	19.36	17.98
fog-0	**94.90**	86.75	91.53	78.87
fog-1	**94.75**	84.65	86.05	65.50
fog-2	**93.98**	79.16	77.93	51.99
fog-3	**91.69**	68.41	64.11	38.62
fog-4	**76.58**	44.17	38.01	24.04

C. Training and Testing on Gaussian Noise

In Section 6, we mentioned that it is not trivial to learn the distribution of noisy images simply by augmenting the training data distribution. In Tables 5 and 6 we present more information about the performance of the models we trained and tested on various scales of Gaussian noise.

D. Results on MNIST

MNIST is a special case when it comes to the relationship between small adversarial perturbations and generalization in noise. Indeed prior has already observed that an MNIST model can trivially become robust to small l_∞ perturbations by learning to threshold the input (Schmidt et al., 2018), and observed that the model from Madry et al. (2017) indeed seems to do this. When we investigated this model in different noise distributions we found it generalizes worse than a naturally trained model, results are shown in Table 7. Given that it is possible for a defense to overfit to a particular l_p metric, future work would be strengthened by demonstrating improved generalization outside the natural data distribution.

Table 5. Wide ResNet-28-10 (Zagoruyko & Komodakis, 2016) trained and tested on CIFAR-10 with Gaussian noise with standard deviation σ.

σ	0.00625	0.0125	0.025	0.075	0.15	0.25
Training Accuracy	100%	100%	100%	100%	99.9%	99.4%
Test Accuracy	96.0%	95.5%	94.8%	90.4%	77.5%	62.2%

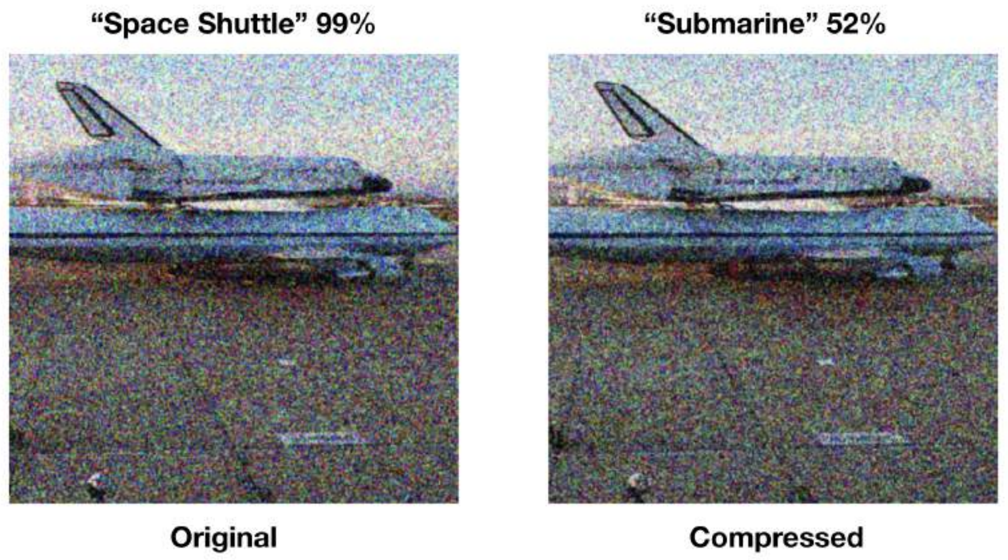

Figure 8. Performance on the ImageNet-C corruptions may vary dramatically depending on whether or not the model is evaluated on the publicly released compressed images vs applying the corruptions directly in memory. For example, an InceptionV3 model trained with Gaussian data augmentation was 57% accurate on the Gaussian-5 corruption when evaluated in memory (example image left). This same model was only 10% accurate on the publicly released compressed images (example image right). The model prediction and confidence on each image is also shown. Note **the image on the right was not modified adversarially**, instead the drop in model performance is due entirely to subtle compression artifacts. This severe degradation in model performance is particularly surprising because differences between the compressed and uncompressed images are difficult to spot for a human. This demonstrates the extreme brittleness of neural networks to *distributional shift*.

Table 6. The models from Section 6 trained and tested on ImageNet with Gaussian noise with standard deviation σ; the column labeled 0 refers to a model trained only on clean images.

σ	0	0.1	0.2	0.4	0.6	0.8
Clean Training Accuracy	91.5%	90.8%	89.9%	87.7%	86.1%	84.6%
Clean Test Accuracy	75.9%	75.5%	75.2%	74.2%	73.3%	72.4%
Noisy Training Accuracy	–	89.0%	85.7%	78.3%	71.7%	65.2%
Noisy Test Accuracy	–	73.9%	70.9%	65.2%	59.7%	54.0%

Here we provide more detail for the noise distributions we used to evaluate the MNIST model. The stAdv attack defines a flow field over the pixels of the image and shifts the pixels according to this flow. The field is parameterized by a latent Z. When we measure accuracy against our randomized variant of this attack, we randomly sample Z from a multivariate Gaussian distribution with standard deviation σ. To implement this attack we used the open sourced code from Xiao et al. (2018). PCA-100 noise first samples noise from a Gaussian distribution $\mathcal{N}(0, \sigma)$, and then projects this noise onto the first 100 PCA components of the data.

E. The Gaussian Isoperimetric Inequality

Here we will discuss the Gaussian isoperimetric inequality more thoroughly than we did in the text. We will present some of the geometric intuition behind the theorem, and in the end we will show how the version quoted in the text follows from the form in which the inequality is usually stated.

The historically earliest version of the isoperimetric inequality, and probably the easiest to understand, is about areas of subsets of the plane and has nothing to do with Gaussians at all. It is concerned with the following problem: among all measurable subsets of the plane with area A, which ones have the smallest possible perimeter?[2] One picture to keep in mind

[2]The name "isoperimetric" comes from a different, but completely equivalent, way of stating the question: among all sets with the

Table 7. The performance of ordinarily and adversarially trained MNIST models on various noise distributions.

Model	Clean Accuracy	Pepper $p = 0.2$ Accuracy	Gaussian $\sigma = 0.3$ Accuracy	stAdv $\sigma = 1.0$ Accuracy	PCA-100 $\sigma = 0.3$ Accuracy
Clean	99.2%	81.4%	96.9%	89.5%	63.3%
Adv	98.4%	27.5%	78.2%	93.2%	47.1%

is to imagine that you are required to fence off some region of the plane with area A and you would like to use as little fence as possible. The isoperimetric inequality says that the sets which are most "efficient" in this sense are balls.

Some care needs to be taken with the definition of the word "perimeter" here — what do we mean by the perimeter of some arbitrary subset of $\mathbb{R}^2$? The definition that we will use involves the concept of the ϵ-boundary measure we discussed in the text. For any set E and any $\epsilon > 0$, recall that we defined the *ϵ-extension* of E, written E_ϵ, to be the set of all points which are within ϵ of a point in E; writing $A(E)$ for the area of E, we then define the perimeter of E to be

$$\text{surf}(E) := \liminf_{\epsilon\to 0} \frac{1}{\epsilon}\left(A(E_\epsilon) - A(E)\right).$$

A good way to convince yourself that this is reasonable is to notice that, for small ϵ, $E_\epsilon - E$ looks like a small band around the perimeter of E with width ϵ. The isoperimetric inequality can then be formally expressed as giving a bound on the quantity inside the limit in terms of what it would be for a ball. (This is slightly stronger than just bounding the perimeter, that is, bounding the limit itself, but this stronger version is still true.) That is, for any measurable set $E \subseteq \mathbb{R}^2$,

$$\frac{1}{\epsilon}(A(E_\epsilon) - A(E)) \geq 2\sqrt{\pi A(E)} + \epsilon\pi.$$

It is a good exercise to check that we have equality here when E is a ball.

There are many generalizations of the isoperimetric inequality. For example, balls are also the subsets in $\mathbb{R}^n$ which have minimal surface area for a given fixed volume, and the corresponding set on the surface of a sphere is a "spherical cap," the set of points inside a circle drawn on the surface of the sphere. The version we are most concerned with in this paper is the generalization to a Gaussian distribution. Rather than trying to relate the volume of E to the volume of E_ϵ, the Gaussian isoperimetric inequality is about the relationship between the *probability* that a random sample from the Gaussian distribution lands in E or E_ϵ. Other than this, though, the question we are trying to answer is the same: for a given probability p, among all sets E for which the probability of landing in E is p, when is the probability of landing in E_ϵ as small as possible?

The Gaussian isoperimetric inequality says that the sets that do this are half spaces. (See Figure 9.) Just as we did in the plane, it is convenient to express this as a bound on the probability of landing in E_ϵ for an arbitrary measurable set E. This can be stated as follows:

Theorem. *Consider the standard normal distribution q on $\mathbb{R}^n$, and let E be a measurable subset of $\mathbb{R}^n$. Write*

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} \exp(x^2/2)dx,$$

the cdf of the one-variable standard normal distribution.

For a measurable subset $E \subseteq \mathbb{R}^n$, write $\alpha(E) = \Phi^{-1}(\mathbb{P}_{x\sim q}[x \in E])$. Then for any $\epsilon \geq 0$,

$$\mathbb{P}_{x\sim p}[d(x, E) \leq \epsilon] \geq \Phi(\alpha(E) + \epsilon).$$

The version we stated in the text involved $\epsilon^*_q(E)$, the median distance from a random sample from q to the closest point in E. This is the same as the smallest ϵ for which $\mathbb{P}_{x\sim p}[d(x, E) \leq \epsilon] = \frac{1}{2}$. So, when $\epsilon = \epsilon^*_q(E)$, the left-hand side of the Gaussian isoperimetric inequality is $\frac{1}{2}$, giving us that $\Phi(\alpha + \epsilon^*_q(E)) \leq \frac{1}{2}$.

Since Φ^{-1} is a strictly increasing function, applying it to both sides preserves the direction of this inequality. But $\Phi^{-1}(\frac{1}{2}) = 0$, so we in fact have that $\epsilon^*_q(E) \leq -\alpha$, which is the statement we wanted.

same fixed perimeter, which ones have the largest possible area?

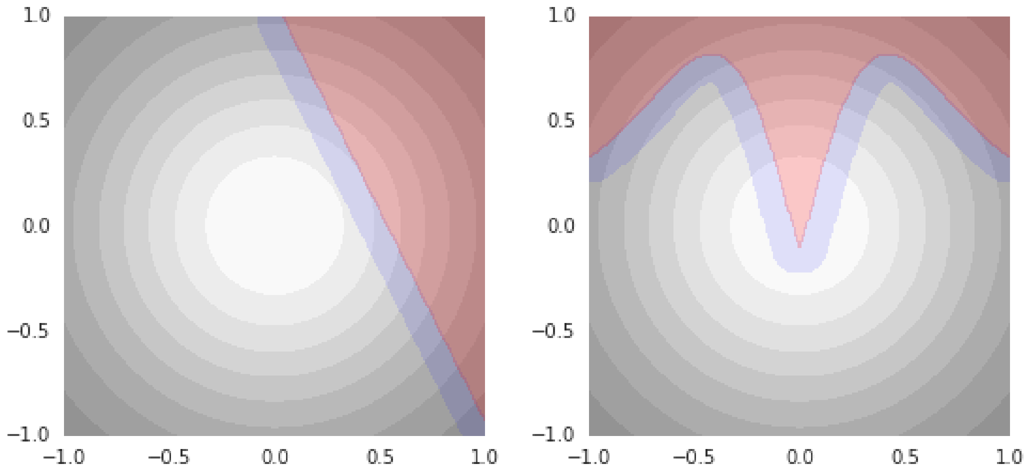

Figure 9. The Gaussian isoperimetric inequality relates the amount of probability mass contained in a set E to the amount contained in its ϵ-extension E_ϵ. A sample from the Gaussian is equally likely to land in the pink set on the left or the pink set on the right, but the set on the right has a larger ϵ-extension. The Gaussian isoperimetric inequality says that the sets with the smallest possible ϵ-extensions are half spaces.

F. Visualizing the Optimal Curves

In this section we visualize the predicted relationship between worst-case l_2 perturbations and generalization in noise as described by Equation 1 in Section 4. This also visualizes the optimal bound according to the isoperimetric inequality, although the l_2 perturbations would be applied to the noisy images themselves rather then clean image. In Figure 10 we plot the optimal curves for various values of σ, visualize images sampled from $x + N(0, \sigma)$, and visualize images at various l_2 distance from the unperturbed clean image. Even for very large noise ($\sigma = .6$), test error needs to be less than 10^{-15} in order to have worst-case perturbations be larger than 5.0. In order to visualize worst-case perturbations at varying l_2 distances, we visualize an image that minimizes similarity according to the SSIM metric (Wang & Bovik, 2009). These images are found by performing gradient descent to minimize the SSIM metric subject to the containt that $||x - x_{adv}||_2 < \epsilon$. This illustrates that achieving significant l_2 adversarial robustness on ImageNet will likely require obtaining a model that is almost perfectly robust to large Gaussian noise (or a model which significantly violates the linearity assumption from Section 4). To achieve l_2 robustness on noisy images, a model *must* be nearly perfect in large Gaussian noise.

G. Church Window Plots

In figures appearing below, starting at Figure 11, we include many more visualizations of the sorts of church window plots we discussed briefly in Section 4. We will show an ordinarily trained model's predictions on several different slices through the same CIFAR test point which illustrate different aspects of the story told in this paper. These images are best viewed in color.

H. The Distribution of Error Rates in Noise

Using some of the models that were trained on noise, we computed, for each image in the CIFAR test set, the probably that a random Gaussian perturbation will be misclassified. A histogram is shown in Figure 20. Note that, even though these models were trained on noise, there are still many errors around most images in the test set. While it would have been possible for the reduced performance in noise to be due to only a few test points, we see clearly that this is not the case.

I. A Collection of Model Errors

Finally, in the figures starting at Figure 21 we first show a collection of iid test errors for the ResNet-50 model on the ImageNet validation set. We also visualize the severity of the different noise distributions considered in this work, along with model errors found by random sampling in these distributions.

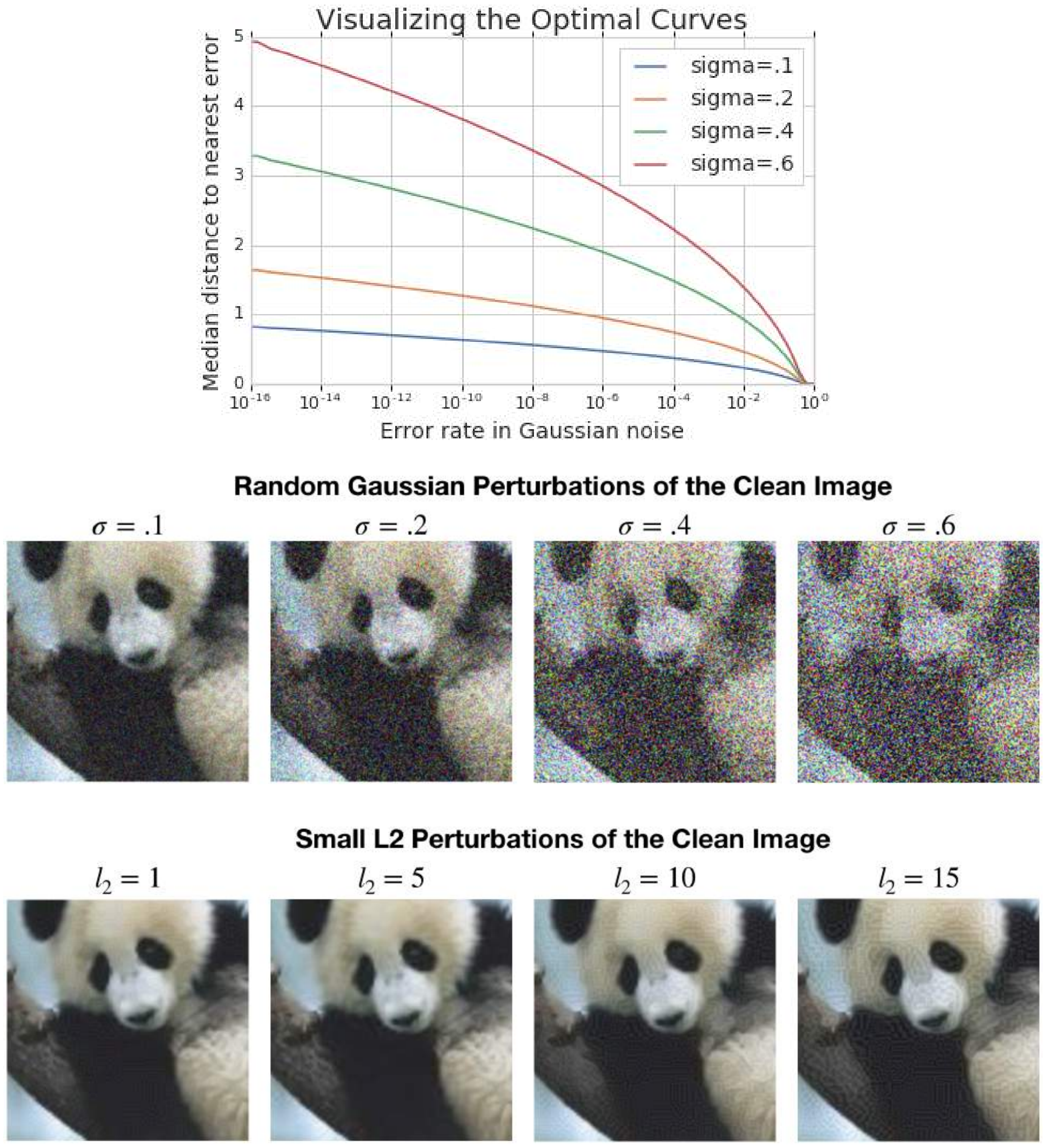

Figure 10. **Top:** The optimal curves on ImageNet for different values of σ. This is both the optimum established by the Gaussian isoperimetric inequality and the relationship described in Equation 1. **Middle:** Visualizing different coordinates of the optimal curves. First, random samples from $x + N(0, \sigma I)$ for different values of σ. **Bottom:** Images at different l_2 distances from the unperturbed clean image. Each image visualized is the image at the given l_2 distance which minimizes visual similarity according to the SSIM metric. Note that images at $l_2 < 5$ have almost no perceptible change from the clean image despite the fact that SSIM visual similarity is minimized.

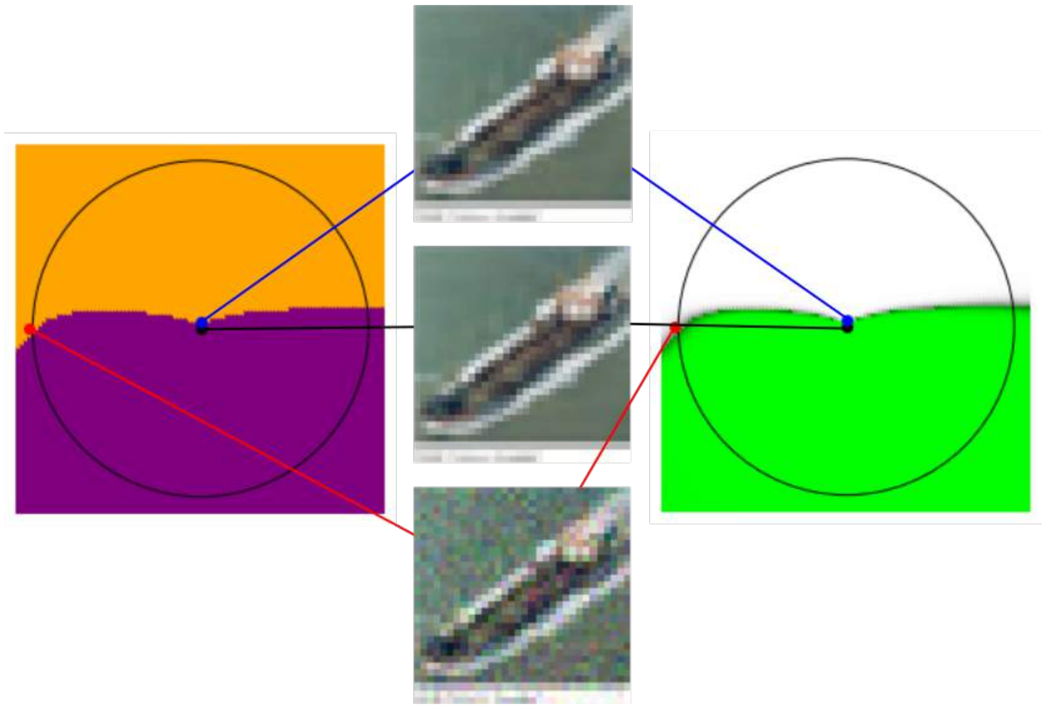

Figure 11. A slice through a clean test point (black, center image), the closest error found using PGD (blue, top image), and a random error found using Gaussian noise (red, bottom image). For this visualization, and all others in this section involving Gaussian noise, we used noise with $\sigma = 0.05$, at which the error rate was about 1.7%. In all of these images, the black circle indicates the distance at which the typical such Gaussian sample will lie. The plot on the right shows the probability that the model assigned to its chosen class. Green indicates a correct prediction, gray or white is an incorrect prediction, and brighter means more confident.

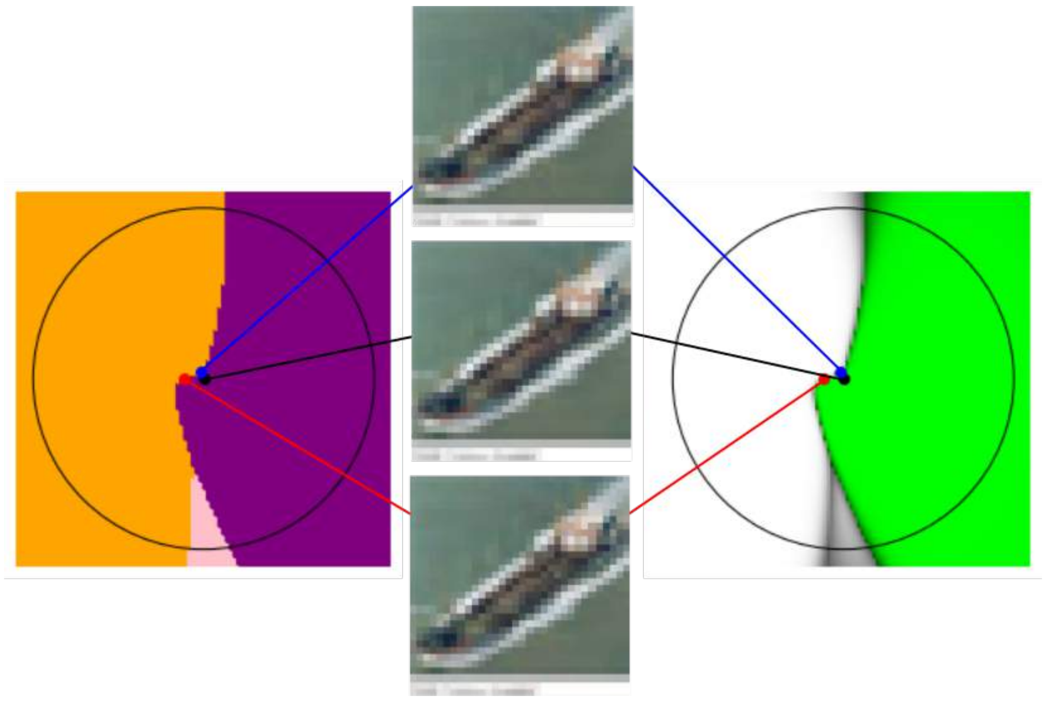

Figure 12. A slice through a clean test point (black, center image), the closest error found using PGD (blue, top image), and the average of a large number of errors randomly found using Gaussian noise (red, bottom image). The distance from the clean image to the PGD error was 0.12, and the distance from the clean image to the averaged error was 0.33. The clean image is assigned the correct class with probability 99.9995% and the average and PGD errors are assigned the incorrect class with probabilities 55.3% and 61.4% respectively. However, it is clear from this image that moving even a small amount into the orange region will increase these latter numbers significantly. For example, the probability assigned to the PGD error can be increased to 99% by moving it further from the clean image in the same direction by a distance of 0.07.

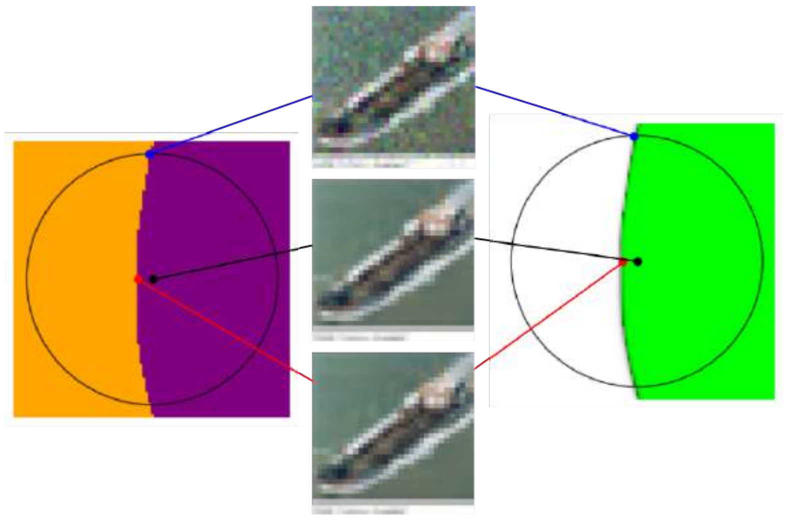

Figure 13. A slice through a clean test point (black, center image), a random error found using Gaussian noise (blue, top image), and the average of a large number of errors randomly found using Gaussian noise (red, bottom image).

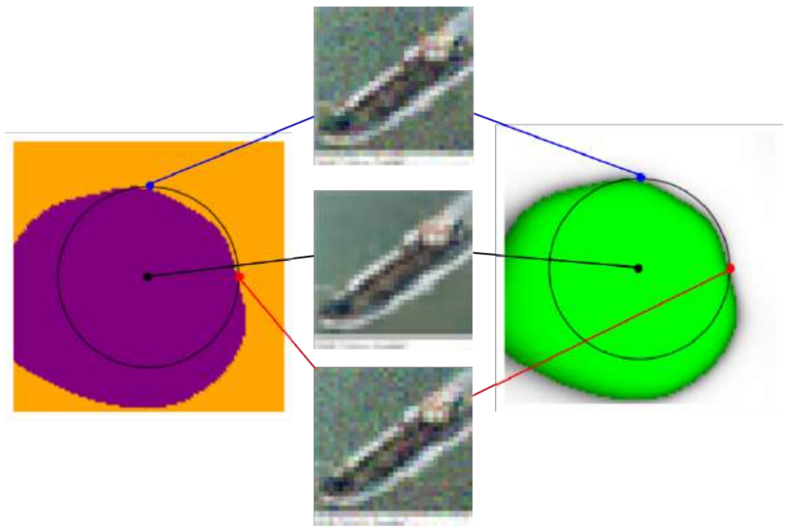

Figure 14. A slice through a clean test point (black, center image) and two random errors found using Gaussian noise (blue and red, top and bottom images). Note that both random errors lie very close to the decision boundary, and in this slice the decision boundary does not appear to come close to the clean image.

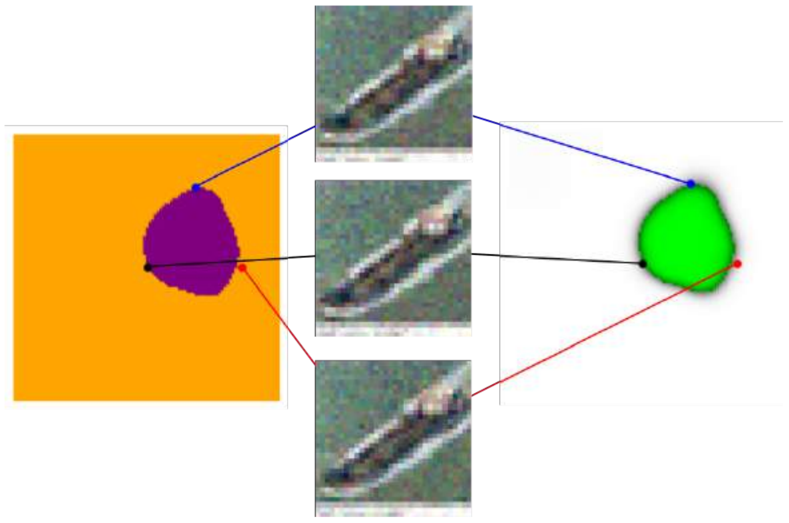

Figure 15. A slice through three random errors found using Gaussian noise. (Note, in particular, that the black point in this visualization does not correspond to the clean image.)

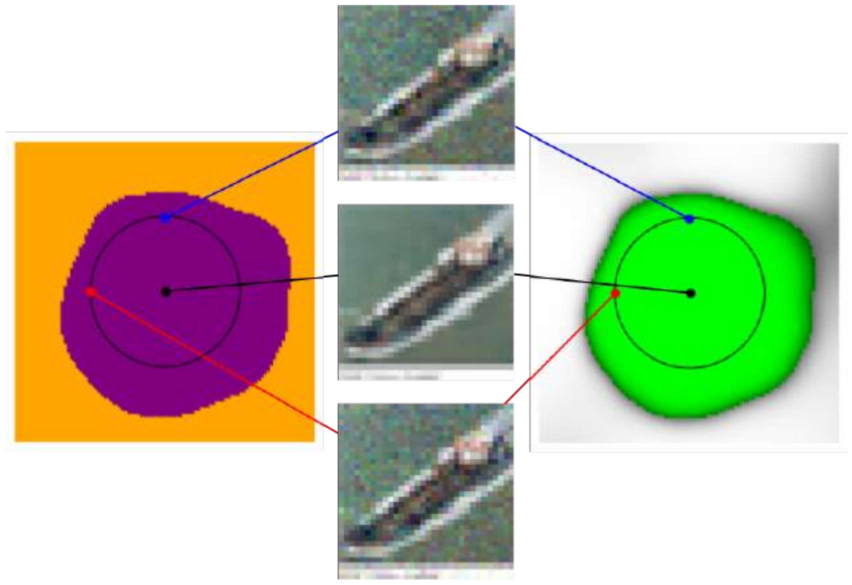

Figure 16. A completely random slice through the clean image.

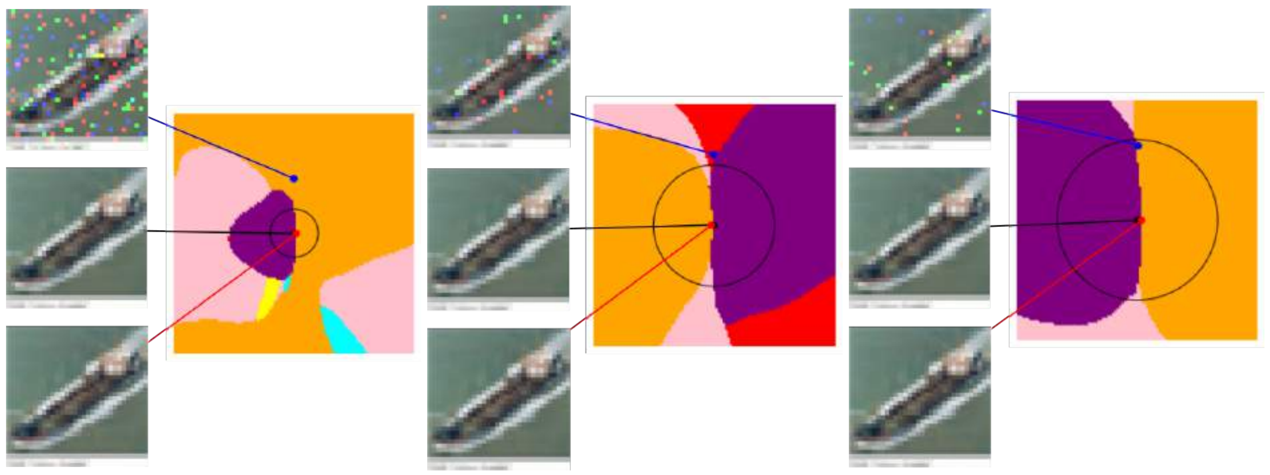

Figure 17. Some visualizations of the same phenomenon, but using pepper noise rather than Gaussian noise. In all of these visualizations, we see the slice through the clean image (black, center image), the same PGD error as above (red, bottom image), and a random error found using pepper noise (blue, top image). In the visualization on the left, we used an amount of noise that places the noisy image further from the clean image than in the Gaussian cases we considered above. In the visualization in the center, we selected a noisy image which was assigned to neither the correct class nor the class of the PGD error. In the visualization on the right, we selected a noisy image which was assigned to the same class as the PGD error.

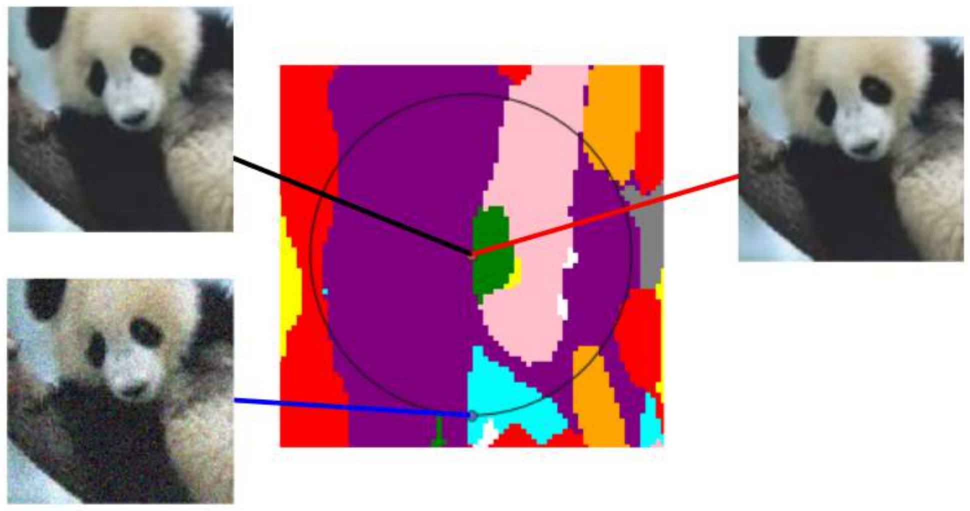

Figure 18. Not all slices containing a PGD error and a random error look like Figure 3. This image shows a different PGD error which is assigned to a different class than the random error.

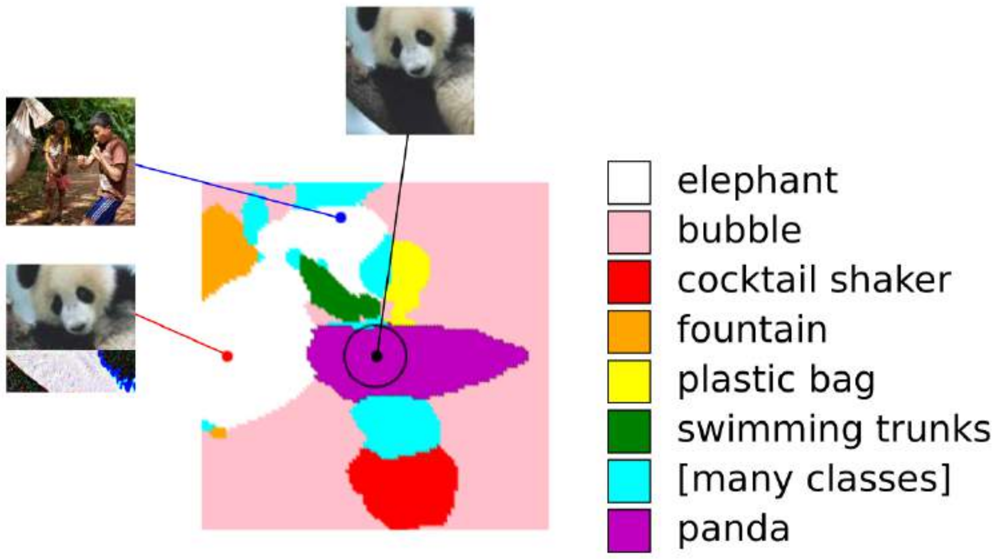

Figure 19. A slice with the same black point as in Figure 3 from the main text, together with an error from the clean set (blue) and an adversarially constructed error (red) which are both assigned to the same class ("elephant"). We see a different slice through the same test point but at a larger scale. This slice includes an ordinary test error along with an adversarial perturbation of the center image constructed with the goal of maintaining visual similarity while having a large l_2 distance. The two errors are both classified (incorrectly) by the model as "elephant." This adversarial error is actually *farther* from the center than the test error, but they still clearly belong to the same connected component. This suggests that defending against worst-case content-preserving perturbations (Gilmer et al., 2018a) requires removing all errors at a scale comparable to the distance between unrelated pairs of images.

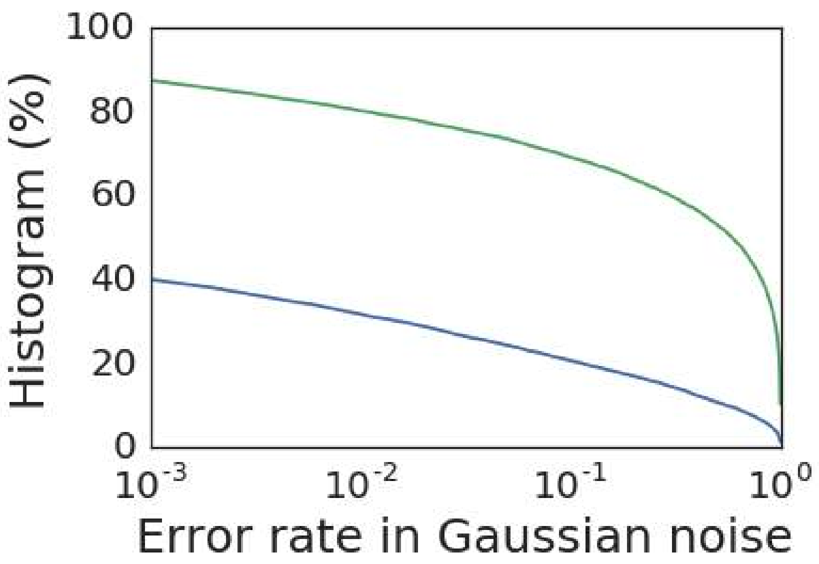

Figure 20. The cdf of the error rates in noise for images in the test set. The blue curve corresponds to a model trained and tested on noise with $\sigma = 0.1$, and the green curve is for a model trained and tested at $\sigma = 0.3$. For example, the left most point on the blue curve indicates that about 40% of test images had an error rate of at least 10^{-3}.

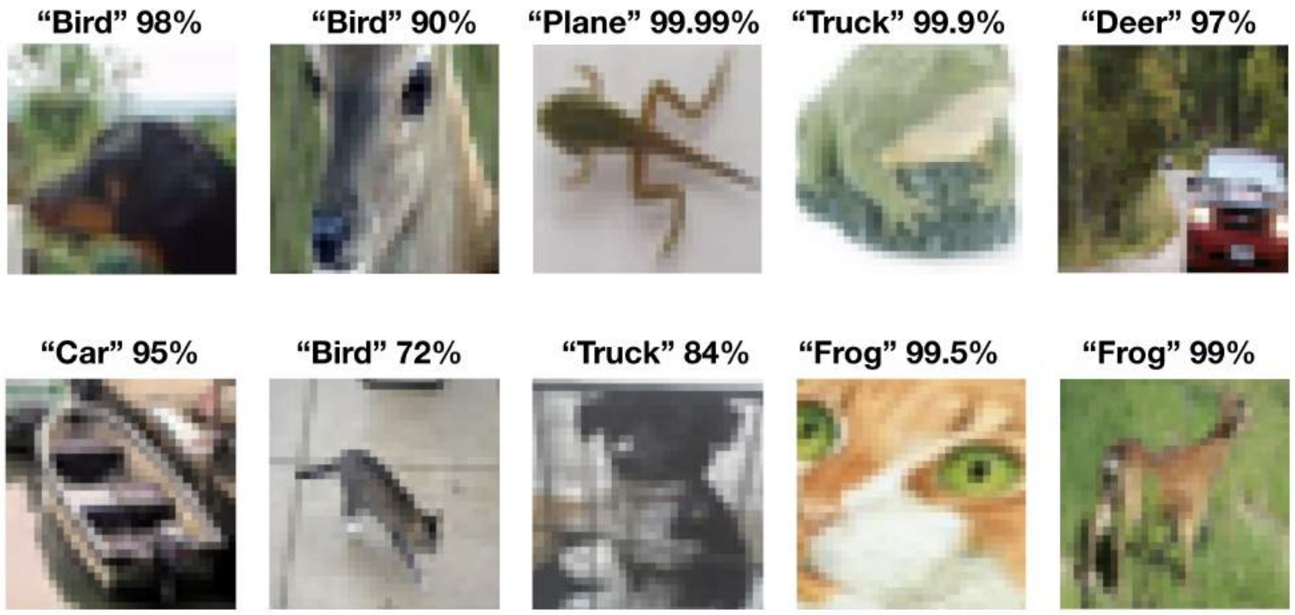

Figure 21. A collection of adversarially chosen model errors. These errors appeared in the ImageNet validation set. Despite the high accuracy of the model there remain plenty of errors in the test set that a human would not make.

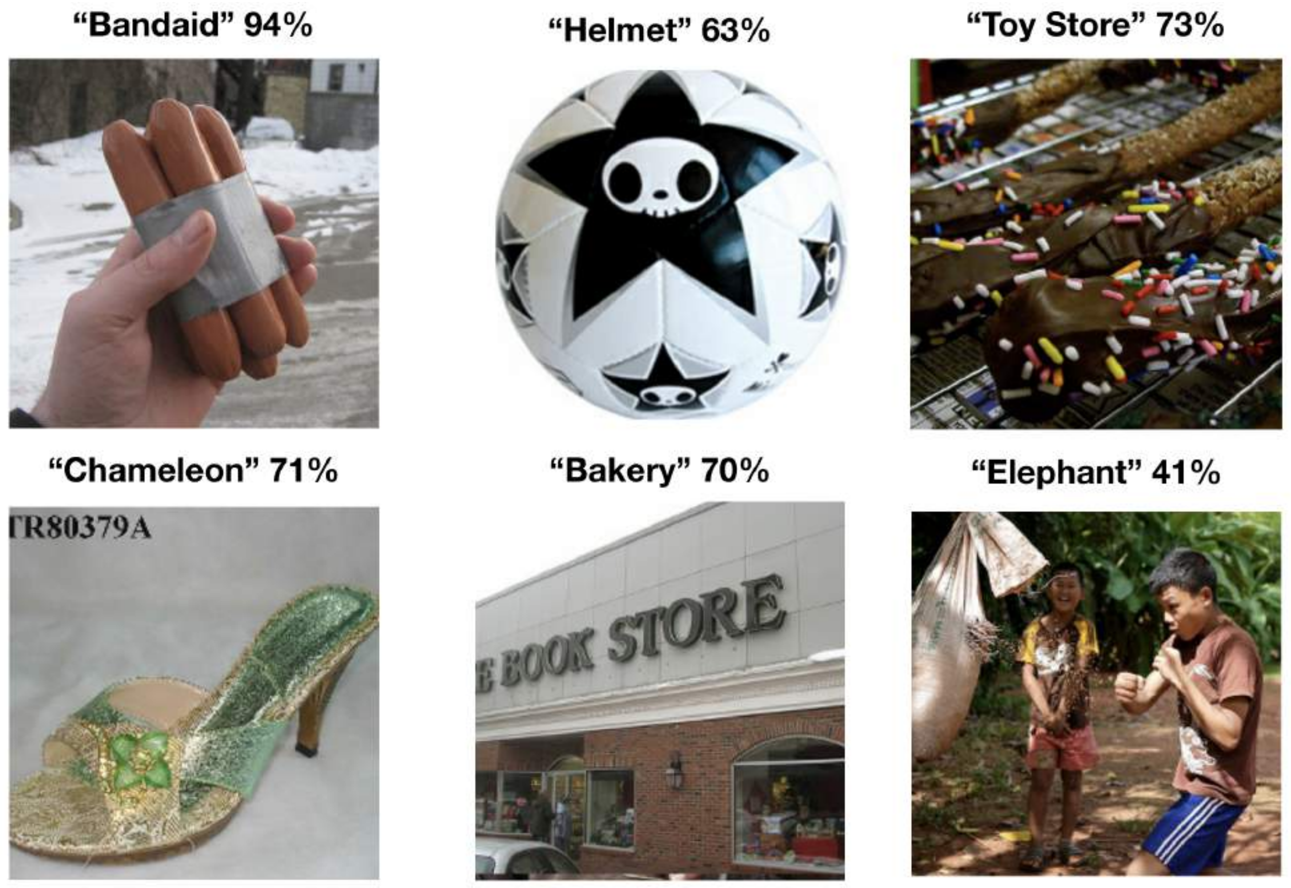

Figure 22. A collection of adversarially chosen model errors. These errors appeared in the ImageNet validation set. Despite the high accuracy of the model there remain plenty of errors in the test set that a human would not make.

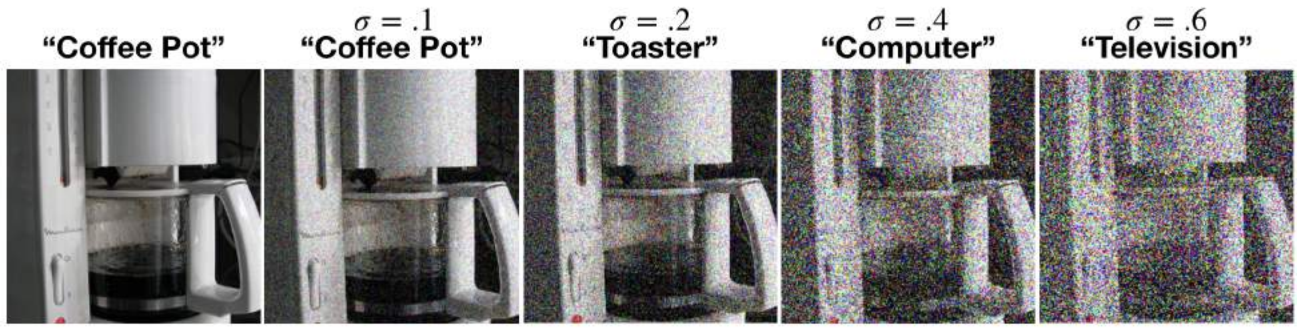

Figure 23. Visualizing the severity of Gaussian noise, along with model errors found in this noise distribution. Note the model shown here was trained at noise level $\sigma = .6$.

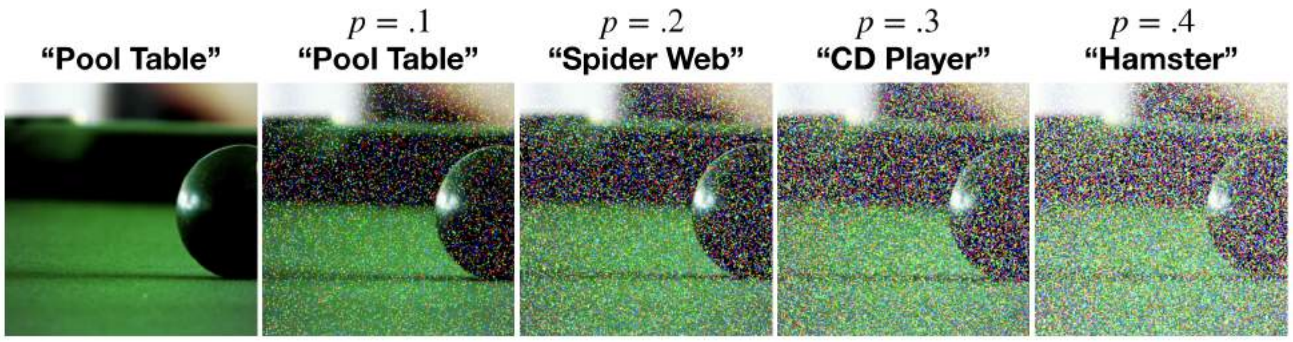

Figure 24. Visualizing the severity of pepper noise.