Localized Uncertainty Attacks

Ousmane Amadou Dia
Facebook
ousamdia@fb.com

Theofanis Karaletsos
Facebook
theokara@fb.com

Caner Hazirbas
Facebook
hazirbas@fb.com

Cristian Canton Ferrer
Facebook
ccanton@fb.com

Ilknur Kaynar Kabul
Facebook
ilknurkabul@fb.com

Erik Meijer
Facebook
erikm@fb.com

Abstract

The susceptibility of deep learning models to adversarial perturbations has stirred renewed attention in adversarial examples resulting in a number of attacks. However, most of these attacks fail to encompass a large spectrum of adversarial perturbations that are imperceptible to humans. In this paper, we present localized uncertainty attacks, a novel class of threat models against deterministic and stochastic classifiers. Under this threat model, we create adversarial examples by perturbing only regions in the inputs where a classifier is uncertain. To find such regions, we utilize the predictive uncertainty of the classifier when the classifier is stochastic or, we learn a surrogate model to amortize the uncertainty when it is deterministic. Unlike ℓ_p ball or functional attacks which perturb inputs indiscriminately, our targeted changes can be less perceptible. When considered under our threat model, these attacks still produce strong adversarial examples; with the examples retaining a greater degree of similarity with the inputs.

1. Introduction

In this paper, we present *localized uncertainty attacks*, a novel class of threat models for creating adversarial examples against deep learning models. Under this threat model, adversarial examples are generated via *replacement* by localizing regions in the original inputs where a classifier is *uncertain or makes unconstrained extrapolations*. By "replacement", we mean that instead of perturbing indiscriminately an input as standard threat models intend to, *only its select regions or points* are subject to adversarial changes. We motivate our attack model in more details in Appendix A.

Unlike standard threat models, which perturb indiscriminately every input, our attack model focuses only on select few pixels; resulting thus in improved imperceptibility while maintaining similar attack success rates. Our perturbation hinges on quantifying the uncertainty associated with the predictions of the target classifiers on specific regions of the inputs. For deterministic classifiers, we train surrogate uncertainty models to get amortized uncertainty estimates. To the authors' knowledge, there is no or limited work on using uncertainty to attack both *deterministic and stochastic* deep learning models. Our threat model can also be coupled with standard threat models to achieve even stronger attacks.

2. Uncertainty Threat Model

We propose localized uncertainty attacks, a novel class of threat models for creating adversarial examples against deep learning models. Under this threat model, adversarial examples are generated via replacement by localizing regions in the input images where a classifier g_θ is poorly constrained.

Framework. In Figure 1, we provide a high-level depiction of our framework. Essentially, our framework consists of a mask model which, when given as input a raw image $\boldsymbol{x}$, learns to locate regions within $\boldsymbol{x}$ where g_θ makes unconstrained extrapolations. The mask model learns a distribution $p_\nu(\boldsymbol{x})$ over binary masks; with each mask $\boldsymbol{\omega}$ delineating regions in $\boldsymbol{x}$ to perturb so as to induce misclassification.

For any given input image $\boldsymbol{x}$, each mask $\boldsymbol{\omega} = (\omega_1, ..., \omega_n)$ is of similar shape as $\boldsymbol{x} = (x_1, ..., x_n)$ and maintains a one-to-one coordinate mapping with $\boldsymbol{x}$. Ideally, we want a mask $\boldsymbol{\omega} \sim p_\nu(\boldsymbol{x})$ which, when applied to $\boldsymbol{x}$ (or $\boldsymbol{\omega} \odot \boldsymbol{x}$), maximizes the predictive uncertainty of g_θ. What this means is that the training of the mask model is based upon maximizing the predictive uncertainty of g_θ. We describe in Section 3.1 how we estimate such uncertainty. Finally, we want the perturbations we apply to $\boldsymbol{\omega} \odot \boldsymbol{x}$ to become adversarial. Understandably, we need to regularize the mask model so that the masks it outputs are neither full, i.e.; $\omega_1 = ... = \omega_n = 1$, nor completely sparse, i.e.; $\omega_1 = ... = \omega_n = 0$.

Uncertainty Threat Model. We consider a *white-box scenario* where we want to craft *strong* and *visually imper-*

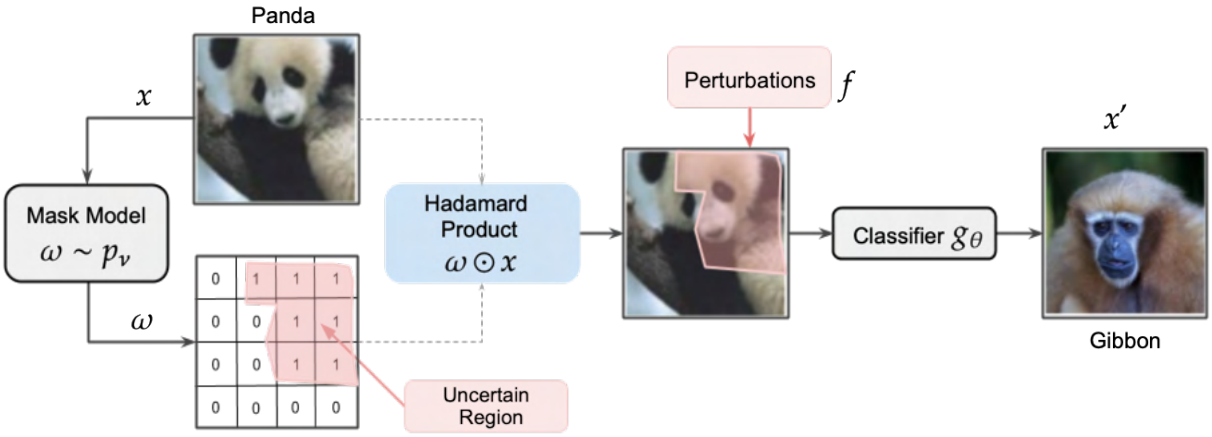


Figure 1: Given the image of a panda as input, the mask model p_ν learns to localize regions in the input where the classifier g_θ is uncertain (or makes unconstrained extrapolations) by outputting a binary mask $\boldsymbol{\omega}$. The pink region in the mask denotes a region in the input where g_θ is poorly constrained. When attacking the input, only the pixels lying in that region will be affected by the perturbation function f.

ceivable adversarial examples by altering as few pixels as possible. We generate adversarial examples by solving:

$$\boldsymbol{x'} = \underset{\boldsymbol{x'} \in \mathcal{S}(f(\boldsymbol{x}, \boldsymbol{\omega}))}{\operatorname{argmax}} \boldsymbol{\mathcal{L}}_{\text{adv}}(\boldsymbol{x'}, \boldsymbol{y}), \text{ where } \boldsymbol{\omega} \sim p_\nu(\boldsymbol{x}) \text{ and}$$
$$f(x_i, \omega_i) = \begin{cases} f(x_i) + \delta_i, & \text{if } \omega_i = 1 \\ x_i, & \text{otherwise} \end{cases} \text{ for } i = 1, ..., n. \tag{1}$$

Here, $\mathcal{S}(\boldsymbol{x})$ denotes the set of permissible perturbations the attacker can inject to $\boldsymbol{x}$ to induce misclassification from g_θ. A common choice for $\mathcal{S}(\boldsymbol{x})$ is an ℓ_p ball of radius ϵ centered on $\boldsymbol{x}$, defined as: $\mathcal{S}(\boldsymbol{x}) = \{\boldsymbol{x} + \boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \leq \epsilon\}$, for some norm $\|.\|_p$ where $p \in \{0, 1, 2, \infty\}$. $\boldsymbol{\mathcal{L}}_{\text{adv}}$ denotes the adversarial attack loss. If g_θ is deterministic, then $\boldsymbol{\mathcal{L}}_{\text{adv}}(\boldsymbol{x'}, \boldsymbol{y}) = \ell(\boldsymbol{x'}, \boldsymbol{y}; \theta)$ else, $\boldsymbol{\mathcal{L}}_{\text{adv}}(\boldsymbol{x'}, \boldsymbol{y}) = \mathbb{E}_{\theta \sim q_\theta} \ell(\boldsymbol{x'}, \boldsymbol{y}; \theta)$ where $q(\theta|\mathcal{D})$ is the predictive posterior over the weights θ.

Our formalism extends gracefully the additive and functional threat models. By defining f as the identity function $f : x \mapsto x$ where $x \in \boldsymbol{\mathcal{X}}$, we recover the additive threat models. By setting $\delta_i = 0$ for all i, we get the functional threat models. Since our threat model extends aforementioned threat models, it inherits their strengths without sacrificing imperceptibility as only few points of $\boldsymbol{x}$ are perturbed.

In what follows, we consider separately the ℓ_p attacks under our threat model (that is; we set f as an identity function in Equation 1), and the functional attacks (that is; $\delta_1 = ... = \delta_n = 0$) under our threat model. When experimenting with the functional attacks, we use the color space transformation function proposed by [24]. Formally, we treat each pixel x_i in the input image $\boldsymbol{x}$ as a point in a 3-dimensional RGB color space, i.e.; $x_i = (c_{i,1}, c_{i,2}, c_{i,3})$ where $c_{i,*} \in [0, 1]$. Then, we use the trilinear interpolation introduced by [24] as the perturbation function f to modify the pixels under the same imperceptibility constraints that they observe for f.

3. Localized Uncertainty Attacks

A prevalent explanation for the existence of adversarial examples is that they lie off the manifold of natural inputs [12, 21], occupying regions where the model makes unconstrained extrapolations [36]. Using this hypothesis as a guiding principle, we could arguably target these regions in order to produce adversarial examples. What this hinges on here is first perturbing parts of an input and measuring the distance of the input to the data manifold. Then, we can quantify the uncertainty associated with the predictions of a target model on the input. One way to measure the uncertainty of a model is to compute the entropy of its predictive distribution.

3.1. Uncertainty Estimation

Taking inspiration from [36], we use entropy in the probability space as a proxy for classification uncertainty. Next, we describe how we measure uncertainty.

Entropy. In the canonical classification setting, where the output of the classifier g_θ is a conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x}; \theta)$ over the discrete set of outcomes $\boldsymbol{\mathcal{Y}}$, the entropy of the predictive distribution of g_θ is defined by:

$$H\left[p(\boldsymbol{y}|\boldsymbol{x}; \theta)\right] = -\sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}} p(\boldsymbol{y}|\boldsymbol{x}; \theta) \log p(\boldsymbol{y}|\boldsymbol{x}; \theta).$$

Mutual Information. According to [36], a good measure of uncertainty that is able to distinguish epistemic from aleatoric examples is the mutual information between the

model parameters and the data, defined as:

$$U_e(\theta, \boldsymbol{y}|\boldsymbol{\mathcal{D}}, \boldsymbol{x}) = \underbrace{H\big[p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mathcal{D}})\big]}_{U_t(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mathcal{D}})} - \underbrace{\mathbb{E}_{p(\theta|\boldsymbol{\mathcal{D}})} H\big[p(\boldsymbol{y}|\boldsymbol{x}; \theta)\big]}_{U_p(\boldsymbol{y}|\boldsymbol{x}, \theta)}. \tag{2}$$

The quantities U_t and U_p are referred to as the predictive entropy and the expected entropy. Mutual information can provide good estimates of a model epistemic uncertainty [36]. In the form presented above, U_e is not practical as the predictive distribution $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mathcal{D}})$ is not analytically tractable. However, $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mathcal{D}})$ can be approximated using the Bayesian interpretation of dropout [38, 16], variational inference [18], or the Mean-Field variational approximation of [4]. In either cases, U_e becomes readily computable using the following:

$$p(\boldsymbol{y}|\boldsymbol{\mathcal{D}}, \boldsymbol{x}) \approx \frac{1}{T} \sum_{t=1}^{T} p(\boldsymbol{y}|\boldsymbol{x}, \theta_t) \text{ where } \theta_t \sim q(\theta|\boldsymbol{\mathcal{D}}).$$

Here, $q(\theta|\boldsymbol{\mathcal{D}})$ denotes the dropout distribution when using dropout variational [16], or the variational approximation to the true posterior $p(\theta|\boldsymbol{\mathcal{D}})$ when using (mean-field) variational inference [4, 18], and θ_t are its samples. With this approximation, we can compute U_e tractably:

$$U_e \approx H\big[\frac{1}{T} \sum_{t=1}^{T} p(\boldsymbol{y}|\boldsymbol{x}, \theta_t)\big] - \frac{1}{T} \sum_{t=1}^{T} H\big[p(\boldsymbol{y}|\boldsymbol{x}, \theta_t)\big]. \tag{3}$$

Amortized Uncertainty. Computing the mutual information U_e in Equation 3 requires g_θ to be stochastic. More formally, in Equation 3 the model weights θ_t which parameterize g_θ are considered as random variables that follow some specific distributions. Essentially, this means that g_θ outputs conditional probabilities rather than point-estimates as it is typically the case for deterministic classifiers. Unfortunately, most of the existing adversarial attacks consider deterministic classifiers. Although we are in a white-box setting where we have complete knowledge about g_θ, we cannot modify g_θ to render its weights stochastic. To circumvent this issue, we learn a surrogate uncertainty model $\mathcal{C}_\phi$ to estimate the quantities U_t and U_p, and compute U_e. We validate empirically these models in Appendix D.

Let $g_{\bar{\theta}}$ denote the embedding sub-network of g_θ, where $\bar{\theta} \subset \theta$. Given an input $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}^n$, let $h = g_{\bar{\theta}}(\boldsymbol{x})$ denote the embedding of $\boldsymbol{x}$ — one can view h as the output we get at the penultimate layer of g_θ. We train $\mathcal{C}_\phi$ by sampling from the Gaussian distribution $\mathcal{N}(h, \text{diag}(\sigma \odot \sigma))$, where the standard deviation σ is outputted by $\mathcal{C}_\phi$ when given h: $\sigma = \mathcal{C}_\phi(h)$. Explicitly, to train $\mathcal{C}_\phi$ we sample M representations $\boldsymbol{z_i} \sim \mathcal{N}(h, \text{diag}(\sigma \odot \sigma))$ for every input $\boldsymbol{x}$. Then, we optimize $\mathcal{C}_\phi$ via cross-entropy using the predictive label distribution of each $\boldsymbol{z_i}$ defined by the multinomial distribution:

$$p(\boldsymbol{y}|\boldsymbol{z_i}; \phi) = \textbf{Softmax}(W^\top \boldsymbol{z_i} + b)[\boldsymbol{y}] \text{ where } \boldsymbol{y} \in \boldsymbol{\mathcal{Y}}.$$

Using $p(\boldsymbol{y}|\boldsymbol{z}; \phi)$, we can estimate U_t and U_p, and then U_e:

$$U_e \approx \underbrace{H\big[\frac{1}{T} \sum_{t=1}^{T} p(\boldsymbol{y}|\boldsymbol{z_t}; \phi)\big]}_{U_t(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\mathcal{D}})} - \underbrace{\frac{1}{T} \sum_{t=1}^{T} H\big[p(\boldsymbol{y}|\boldsymbol{z_t}; \phi)\big]}_{U_p(\boldsymbol{y}|\boldsymbol{x}, \theta)}, \tag{4}$$

where $\boldsymbol{z_t} \sim \mathcal{N}(h, \text{diag}(\sigma \odot \sigma))$, $h = g_{\bar{\theta}}(\boldsymbol{x})$ and $\sigma = \mathcal{C}_\phi(h)$. To motivate the intuition behind $\mathcal{C}_\phi$, we use $g_{\bar{\theta}}$ to embed $\boldsymbol{x}$ into the low-dimensional predictive mean h, which we then feed to $\mathcal{C}_\phi$ to get the predictive standard error σ. Here, $g_{\bar{\theta}}$ learns the predictive features of $\boldsymbol{x}$ while $\mathcal{C}_\phi$ captures the uncertainty associated with their predictiveness. As such, we can use h and σ to parameterize a normal distribution and get amortized estimates of the predictive uncertainty.

3.2. Generating Adversarial Examples

To generate adversarial examples using our uncertainty threat model, we optimize the objective $\boldsymbol{\mathcal{L}}$ defined below:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y}) \triangleq \max_{\boldsymbol{\omega} \sim p_\nu(\boldsymbol{x})} \Big(U_e(\star, \boldsymbol{y}|\boldsymbol{\mathcal{D}}, \boldsymbol{\omega} \odot \boldsymbol{x}) + \max_{\boldsymbol{x'} \in \mathcal{S}(f(\boldsymbol{x}, \boldsymbol{\omega}))} \boldsymbol{\mathcal{L}}_{\text{adv}}(\boldsymbol{x'}, \boldsymbol{y}) - \lambda \cdot \|\boldsymbol{\omega}\|_1 \Big). \tag{5}$$

$U_e(\star, \boldsymbol{y}|\boldsymbol{\mathcal{D}}, \boldsymbol{\omega} \odot \boldsymbol{x})$ is the epistemic uncertainty on $\boldsymbol{\omega} \odot \boldsymbol{x}$ where $\star$ denotes either θ if the classifier g_θ is stochastic, or ϕ if g_θ is deterministic. To compute $U_e(\star, \boldsymbol{y}|\boldsymbol{\mathcal{D}}, \boldsymbol{\omega} \odot \boldsymbol{x})$, we can replace $\boldsymbol{x}$ by $\boldsymbol{\omega} \odot \boldsymbol{x}$ either in Equation 3 or Equation 4. $\boldsymbol{\mathcal{L}}_{\text{adv}}$ is defined in Equation 1. Here, p_ν denotes the distribution over all possible masks $\boldsymbol{\omega}$. $\mathcal{S}(f(\boldsymbol{x}, \boldsymbol{\omega}))$, defined in Equation 1, denotes the set of permissible perturbations we can apply to $\boldsymbol{x}$ in order to induce misclassification from g_θ.

To learn the distribution p_ν, we train a feedforward network called mask model. This network takes as input an image $\boldsymbol{x}$ and outputs probabilities for each point in $\boldsymbol{x}$ (across all its channels). We use the probabilities to parameterize a Bernoulli distribution from which we then sample binary masks. A mask $\boldsymbol{\omega} \sim p_\nu(\boldsymbol{x})$ is of the same size as $\boldsymbol{x}$ with a one-to-one coordinate mapping with $\boldsymbol{x}$. We regularize the mask model so that $\boldsymbol{\omega}$ is neither full, that is; $\omega_1 = ... = \omega_n = 1$, nor completely sparse, i.e.; $\omega_1 = ... = \omega_n = 0$. To that end, we add a sparsity penalty $\|\boldsymbol{\omega}\|_1$ weighted by λ to control its effect on the remaining part of the objective $\boldsymbol{\mathcal{L}}$.

4. Experiments & Results

We wish to show that under our threat model both ℓ_p and functional attacks can retain similar attack strengths while perturbing the inputs less than under their own threat model. In that regard, we seek to answer the following questions through our experiments: (Q1) *How strong are the attacks?* (Q2) *How imperceivable are the adversarial examples?* (Q3) *How transferable are the attacks?* Thus, we validate the

Table 1: Accuracy of defended classifiers before and after *delta* and ReColor attacks, under their natural threat model and ours.

	Attacks ↓	*Deterministic Defenses* (%)			*Bayesian Defenses* (%)	
		None	*delta*	*StdAdv*	None	*delta*
MNIST	Vanilla	98.89	98.64	97.33	97.91	76.76
	delta	**0.00**	**92.47**	**0.00**	**0.00**	**0.00**
	Unc. + *delta*	**0.00**	94.79	**0.00**	**0.00**	**0.00**
CIFAR-10	Vanilla	92.20	84.80	82.88	89.62	77.96
	delta	**0.00**	**30.60**	**0.00**	**0.00**	**1.38**
	Unc. + *delta*	**0.00**	33.32	**0.00**	0.13	3.26
	ReColor	**63.76**	**75.12**	**68.62**	**83.40**	**69.58**
	Unc. + ReColor	65.69	78.64	**68.62**	85.19	71.18
STL-10	Vanilla	81.37	63.42	**N/A**	74.85	59.89
	delta	**0.00**	**0.00**	**N/A**	**0.00**	**0.66**
	Unc. + *delta*	**0.00**	**0.00**	**N/A**	**0.00**	1.44
	ReColor	**53.51**	**44.72**	**N/A**	**15.25**	**18.55**
	Unc. + ReColor	57.23	45.26	**N/A**	17.07	21.03

Table 2: Mask Sparsity (ℓ_0). For vanilla *delta* and ReColor, we get the score by computing the ℓ_0 norm between an input and its adversarial example divided by $C \times W \times H$ where C is the number of channels, W the width and H the height of the images. **Lower is better.**

	Attacks ↓	*Deterministic Defenses* (%)			*Bayesian Defenses* (%)	
		None	*delta*	*StdAdv*	None	*delta*
MNIST	*delta*	88.61	98.73	93.11	88.83	89.20
	Unc. + *delta*	**74.92**	**79.88**	**73.11**	**69.82**	**75.98**
CIFAR-10	*delta*	86.92	78.61	89.33	99.24	99.70
	Unc. + *delta*	**55.40**	**55.68**	**86.97**	**62.66**	**62.65**
	ReColor	99.55	100.0	100.0	100.0	100.0
	Unc. + ReColor	**55.75**	**55.78**	**89.21**	**46.42**	**46.35**
STL-10	*delta*	43.38	75.82	**N/A**	99.43	100.0
	Unc. + *delta*	**28.14**	**27.17**	**N/A**	**43.89**	**58.93**
	ReColor	99.23	100.0	**N/A**	100.0	100.0
	Unc. + ReColor	**78.12**	**58.40**	**N/A**	**58.93**	**58.93**

attacks under our threat model based on three criteria: *attack success rate, imperceptibility, and transferability.*

Datasets. Taking inspiration from similar studies, we conduct experiments on CIFAR-10 [23], MNIST [26], and STL-10 [10]. For a detailed overview of the experimental setup and more results, we refer the reader to Appendix C and D.

Attack Success Rate. We define *attack success rate* as the decrease in test accuracy of a classifier resulting from the misclassifications that our perturbations induce. Here, we ask ourselves the following: *Can delta or ReColor retain their attack success rates under our threat model while perturbing the inputs much less?* To answer this question, we evaluate the sparsity of the binary masks we generate using the ℓ_0 *norm as a sparsity measure*.

We report in Table 1 the attack success rates of *delta* and ReColor attacks under their threat model and under ours. The results in Table 1 are to be read side by side with the sparsity scores in Table 2. Vanilla denotes the targeted classifier with None reflecting its accuracy when it is undefended. *delta* and *StdAdv* show the drop in accuracy after attacking Vanilla when defended either with *delta* or *StdAdv*. As the results show, *delta* and ReColor achieve similar attack success rates under our threat model while perturbing the inputs consistently less than under their own. We explore in Appendix the effects of the mask models on the attacks.

Both *delta* and ReColor achieve better imperceptibility under our threat model than under their native ones, as evidenced by our results in Table 4 in Appendix D. In Table 5 in Appendix D, we also show that the reduced number of perturbed pixels do not seem to significantly impact the transferability

of the generated adversarial examples.

5. Conclusion

We presented *uncertainty attacks*, a novel class of adversarial attacks against deep learning models. Our attack model utilizes the uncertainty associated with the predictions of a stochastic classifier or the amortized uncertainty of its surrogate to localize regions in the inputs to perturb so as to induce misclassification. Unlike standard attacks which perturb inputs indiscriminately, we focus only on carefully chosen regions, yielding thus improved imperceptibility. Mainstream attacks can still be evaluated under our threat model, both against defended and undefended classifiers, and produce strong adversarial examples that retain actually a greater degree of similarity with the original inputs.

References

[1] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, 2018. 7

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. 7

[3] Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and D. A. Forsyth. Unrestricted adversarial examples via semantic manipulation, 2020. 8

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015. 3, 9

[5] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017. 7

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 7

[7] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks, 2020. 9

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society. 7, 8, 9

[9] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models, 2017. 7

[10] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. 4

[11] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *CoRR*, abs/1907.02044, 2019. 7

[12] Ousmane Amadou Dia, Elnaz Barshan, and Reza Babanezhad. Semantics Preserving Adversarial Learning. *arXiv e-prints*, page arXiv:1903.03905, Mar. 2019. 2, 7

[13] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. 7

[14] Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both human and computer vision. *CoRR*, abs/1802.08195, 2018. 7, 8

[15] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, abs/1712.02779, 2017. 7, 8, 9

[16] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv e-prints*, page arXiv:1506.02142, June 2015. 3

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harvesting adversarial examples. *CoRR abs/1412.6572*, 2014. 7, 12

[18] Alex Graves. Practical variational inference for neural networks. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2348–2356, Red Hook, NY, USA, 2011. Curran Associates Inc. 3

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 9

[20] Matt Jordan, Naren Manoj, Surbhi Goel, and Alexand ros G. Dimakis. Quantifying Perceptual Distortion of Adversarial Examples. *arXiv e-prints*, page arXiv:1902.08265, Feb. 2019. 7, 8

[21] Marc Khoury and Dylan Hadfield-Menell. On the geometry of adversarial examples. *CoRR*, abs/1811.00525, 2018. 2

[22] J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017. 7

[23] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research), 2009. 4

[24] Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. *CoRR*, abs/1906.00001, 2019. 2, 7, 8, 9

[25] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 9

[26] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. 4

[27] Chen Liu, Ryota Tomioka, and Volkan Cevher. On certifying non-uniform bound against adversarial attacks. *CoRR*, abs/1903.06603, 2019. 7

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. 7, 9

[29] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016. 12

[30] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018. 7

[31] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *CoRR*, abs/1808.05665, 2018. 7

[32] Andrew Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577:1–5, 01 2020. 7

[33] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. On the suitability of lp-norms for creating and preventing adversarial examples. *CoRR*, abs/1802.09653, 2018. 7

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 9

[35] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. 7

[36] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection, 2018. 2, 3, 9

[37] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models, 2018. 7, 8

[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 3

[39] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-Calibrated Adversarial Training: Generalizing to Unseen Attacks. *arXiv e-prints*, page arXiv:1910.06259, Oct. 2019. 7

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 7, 12

[41] Rey Reza Wiyatno, Anqi Xu, Ousmane Dia, and Archy de Berker. Adversarial examples in modern machine learning: A review. *CoRR*, abs/1911.05268, 2019. 7, 8

[42] Eric Wong and J. Zico Kolter. Learning perturbation sets for robust machine learning, 2020. 8

[43] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *CoRR*, abs/1801.02612, 2018. 9

[44] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review, 2019. 7, 8

[45] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019. 8

[46] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. *CoRR*, abs/1907.10764, 2019. 7

[47] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CoRR*, abs/1801.03924, 2018. 11

A. Motivations

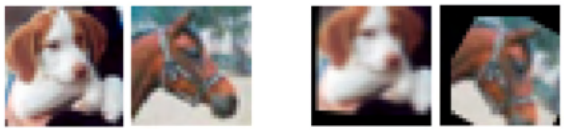
Deep learning models continue to achieve impressive performances on a variety of challenging machine learning tasks [32, 6]. However, they remain fragile. Small, seemingly inconspicuous noise injected to the input data can cause a deep learning model to suddenly make erroneous predictions with high confidence [17, 40]. As deep learning models permeate different and vital segments of the technology industry, such susceptibility to adversarial perturbations is a cause for concern and calls upon machine learning practitioners to revisit their evaluation protocol.

Recently, many methods have been proposed to scrutinize deep learning models and expose their weaknesses in areas as diverse as computer vision [14, 5], speech recognition [31, 9], and machine translation [13].[1] These methods known as adversarial attacks aim generally at producing adversarial examples that can mislead a model while being perceptually similar to the original inputs [24, 12, 11, 28, 20, 33, 2, 40].

By and large, most of the adversarial attacks [1, 15, 8] and defense methods [46, 27, 35, 30, 22, 17] consider *additive ℓ_p threat models* where adversarially crafted examples and their natural inputs differ by a small ℓ_p distance. Concretely, the degree to which an adversarial example is perceptually similar to its original input is measured in terms of an ℓ_p distance as a substitute proxy of human perception of similarity.

However, it is well documented that *nearness* according to an ℓ_p norm is unsuitable for measuring perceptual similarity when creating adversarial examples [33]. In the image domain specifically, clipping, rotation, occlusions, changes of color or illumination in an image, to name a few, do not always preserve the true nature of the image [33]. *Oftentimes, the resulting images appear perceptually distorted, blurry, or unrealistic* as exemplified in Figure 2. In Figure 3, we can also notice two adversarial images that appear similar to their original inputs, yet their ℓ_p distances are — $\ell_0 \geq 3, 10$, $\ell_2 \geq 15.83$, and $\ell_\infty \geq 0.87$ [33] — much higher than the ℓ_p norm thresholds used in most of the existing attacks.

Figure 2: These adversarial images are supposed to represent the digits 0 and 1, yet they look heavily distorted, blurry, and not quite legitimate 0s or 1s. The images are taken from [37].



(a) Original images (b) Adversarial examples

Figure 3: Adversarial examples created using geometric distortions may look perceptually to the inputs but they have large ℓ_p distances in the input space. Images taken from [15].

Although quantifying imperceptibility can be challenging due to the subjective nature of human perception, just using ℓ_p norms as proxies may hinder the construction of adversarial examples that preserve the key features of the inputs. Such an outcome is not only desirable for probing deep learning models in order to detect their weaknesses, it is equally important for increasing their robustness to such attacks. Indeed, many defenses underlied by the ℓ_p threat models tend to be brittle [1]. When they appear robust, their robustness does not always generalize to larger yet similar ℓ_p perturbations, nor does it extrapolate to different threat models [39]. A failure to investigate more inconspicuous attacks could cause major impediments in deploying and entrusting deep learning models with making decisions in security-sensitive applications.

In this paper, we present *localized uncertainty attacks*, a novel class of threat models for creating adversarial examples against deep learning models. Under this threat model, adversarial examples are generated via *replacement* by localizing regions in the original inputs where a classifier is *uncertain or makes unconstrained extrapolations*. By "replacement", we mean that instead of perturbing indiscriminately an input as standard threat models intend to, *only select regions or points* in the input are subject to adversarial changes. Our attack model can essentially be decomposed into: *region proposals* and *adversarial perturbations*.

[1] For a detailed review of the adversarial methods, we refer the interested reader to [41] and [44].

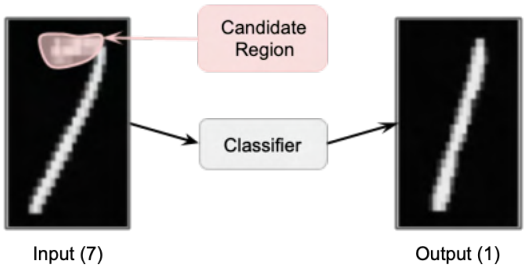


Figure 4: On the left we have a digit 7 to which is overlaid a region proposal colored in pink. Grayscaling this pink region can induce misclassification as the classifier may interpret the input as digit 1.

Region Proposals. Given an input, we learn which regions in the input to perturb. *A region is a set of points which may or may not cluster together*. For instance, a region could denote a patch or a set of non-adjacent pixels in an image, filterbank energies in an audio sample, or a set of words in a sentence. We prioritize mainly the regions where the target classifier is poorly constrained. For illustration, we give a high-level depiction of such a region in Figure 4. To select which regions to perturb, we learn a model which when given the input produces a *binary mask* of similar shape as the input. The mask maintains a one-to-one coordinate mapping with the input. Explicitly, every element of the mask is mapped to the same coordinate point in the input and dictates whether to perturb that point or to leave it unaltered.

Adversarial Perturbations. Our uncertainty threat model can be applied to a wide variety of spatio-temporal modalities including images, text, combination thereof (e.g., OCR), or speech. In this paper, however, we focus only on images. Our threat model extends additive and functional threat models. For instance, we can leverage the losses in [8] to perturb input images. Similar to [24], we can perturb pixels using parameterized functions to transform their color space uniformly in order to induce misclassification. Unlike these threat models, however, we do not perturb arbitrarily every pixel in an image. *We perturb only the select few pixels that our masking algorithm recommends. As a result, we achieve improved imperceptibility over additive and functional attacks*.

For evaluation, we consider additive and functional attacks under our threat model and target undefended and defended classifiers — which are either deterministic or stochastic. We find that under our threat model both attacks are strong attacks, lowering the accuracy of a ResNet-32 trained on CIFAR-10 to 0.0% while perturbing 31% less than under a typical ℓ_p threat model. The combination reduces in one instance the accuracy of an STL-10 classifier to 0.0% while perturbing 43% less than under a standard ℓ_p threat model.

Contributions. To summarize our contributions, we propose localized uncertainty attacks, a novel class of threat models for creating adversarial examples in the image domain. Unlike standard threat models, which perturb indiscriminately every input, our attack model focuses only on select few pixels; resulting thus in improved imperceptibility while maintaining similar attack success rates. Our perturbation hinges on quantifying the uncertainty associated with the predictions of the target classifiers on specific regions of the inputs. For deterministic classifiers, we train surrogate uncertainty models to get amortized uncertainty estimates. To the authors' knowledge, there is no or limited work on using uncertainty to attack both *deterministic and stochastic* deep learning models. Our threat model can also be coupled with standard threat models to achieve even stronger attacks.

B. Related Work

In this section, we review some of the current studies that are most closely related to our approach. For a more detailed review of adversarial attacks, we refer the interested reader to [41, 44, 45].

Adversarial Examples have received in recent years renewed interests resulting in a number of approaches for generating novel adversarial attacks and defending against them. Many adversarial attacks focus on finding small, inconspicuous perturbations that can deceive deep learning models while being imperceptible to humans [42, 20, 14]. However, several other notions of adversarial perturbations have been studied [3, 15, 37]. Our approach falls in the category of attacks that look to craft adversarial examples with minimal distortion to the inputs. In this study, we proposed a new method to improve the imperceptibility of adversarial perturbations by carefully selecting which regions in an input to perturb.

Uncertainty Estimation. To select which regions in an input to perturb, we localize the regions where a classifier appears

uncertain or poorly constrained. Taking inspiration from [36], we use entropy in the probability space as a proxy for classification uncertainty. Using this proxy, we can retrieve the epistemic uncertainty of a stochastic classifier and perform gradient-based attacks. We extend this approach of quantifying epistemic uncertainty to deterministic classifiers using surrogate models to get amortized uncertainty estimates with minimal computational overhead.

[7] have argued that Bayesian Neural Networks are robust to gradient-based attacks. However, the conditions under which their findings hold are hard to observe in practice. To the author's knowledge, there is no or limited study that uses uncertainty to perform gradient-based attacks against both deterministic and stochastic deep learning models. Owing to its simplicity, our attack model can be easily combined with a variety of attack models including *delta* [28, 8], functional [24], and spatial transformation attacks [43, 15] to minimize perceptual distortions without sacrificing strength.

C. Experimental Setup

Experimental Setup. To validate our uncertainty threat model, we consider deterministic and stochastic classifiers.

Deterministic Classifiers. For CIFAR-10, we consider an undefended ResNet-32 [19], a ResNet-32 robustified with *delta* attacks, and a ResNet-32 defended with StdAdv [43]. We use the classifiers pretrained by [24]. For MNIST, we evaluate with an undefended LeNet5 [25], a LeNet5 trained with StdAdv, and a LeNet5 trained with PGD [28]. For STL-10, we experiment with the undefended and defended classifiers pretrained by [43].

For each of the deterministic classifiers, we train a surrogate uncertainty model to get amortized uncertainty estimates which we then use to train a separate mask model. Note that the surrogate model and the mask model are trained jointly. The surrogate model is a feedforward network trained with dropout. The mask model is an hourglass model consisting of convolutional layers. We use a reparameterized Tanh over the outputs of the mask model to define a (relaxed) Bernoulli distribution from which we sample the binary masks $\boldsymbol{\omega}$.

Stochastic Classifiers. We experiment with MNIST, CIFAR-10 and STL-10 where we evaluate our attack model against Bayesian classifiers. For MNIST, we experiment with an undefended Bayesian LeNet5 that we train using mean-field variational inference [4], and one defended with PGD. For CIFAR-10, we consider two Bayesian VGGs [34]: an undefended VGG and one defended with ADV-BNN [43] both pretrained by [43]. For STL-10, we experiment with the models provided by [43].

Perturbations. Under our threat model, we evaluate the *delta* attacks which use an ℓ_∞ additive threat model with $\epsilon = 8/255$. Under also our threat model, we evaluate the functional attacks [24] which transform uniformly the color space of all the pixels of an image. In what follows, *delta* refers to the additive perturbations, and ReColor to the trilinear transformation of [24] in the RGB color space where we relax their smoothness constraint. For the latter, we allow each channel of a color to change by $8/255$ at most, that is; $\epsilon_R = \epsilon_G = \epsilon_B = 0.03$. We use Projected Gradient Descent or PGD [28] with 100 iterations to carry out the uncertainty attacks. Finally, we consider as adversarial loss the f_6 loss from [8] with $\kappa = 10$.

D. More Evaluation Results

D.1. Surrogate Models

To attack deterministic classifiers, we train surrogate models to get amortized uncertainty estimates. A surrogate model is two-layers deep and is trained jointly with the mask model to minimize overhead. To validate empirically these models, we compare in Figure 5 a Mean-Field with the surrogate of a deterministic LeNet5 based on their predictive probability and entropy on hold-out MNIST images and their adversaries. As Figure 5 shows, the surrogate approximates well the Mean-Field on both the inputs and their adversaries.

We extend this empirical validation further using KL divergence conditioned on the predictive distribution of the deterministic LeNet5 to measure the discrepancy between both models based on their distributions. Formally, if we denote the predictive distributions of the deterministic LeNet5, the surrogate model, and the Mean-field respectively by P_d, P_s, and P_m, we compare the surrogate and the Mean-field based on the divergences $D_{KL}(P_d \| P_m)$ and $D_{KL}(P_s \| P_m)$ both on the natural MNIST images and their adversarial examples. We show in Figure 6 the kernel density estimation plots for both divergences and summarize the aggregated KL in the table nested in Figure 6. As we can notice, here too the surrogate model approximates quite well the Mean-field model as $D_{KL}(P_s \| P_m)$ is much smaller than $D_{KL}(P_d \| P_m)$.

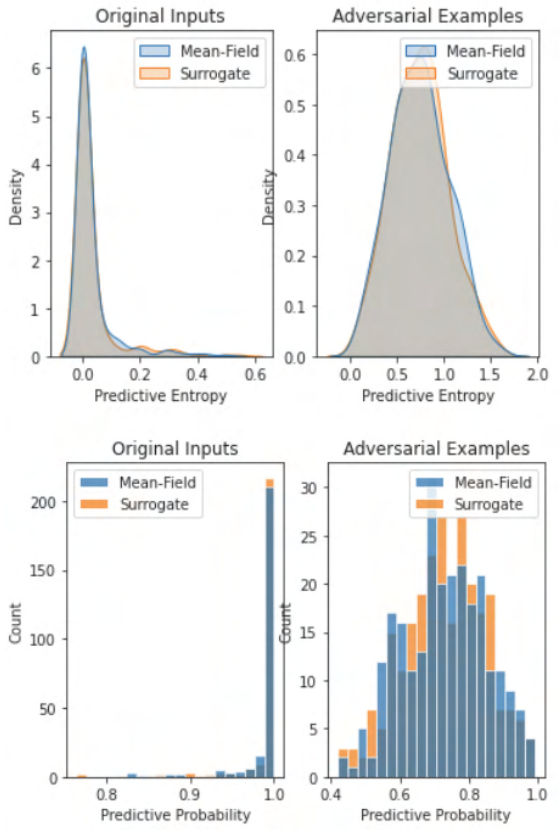
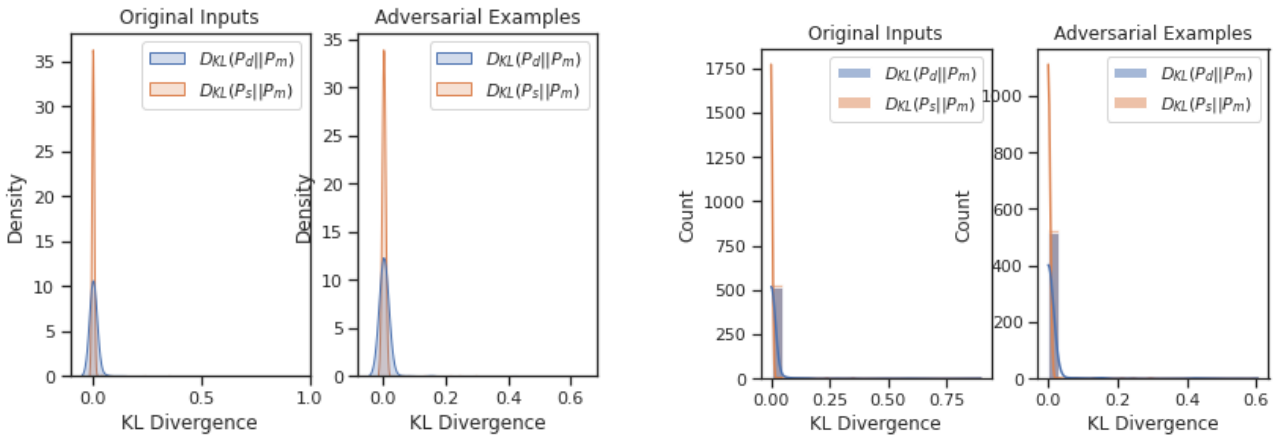


Figure 5: Mean-Field model compared to a LeNet surrogate.



$D_{KL}(P_d\|P_m)$		$D_{KL}(P_s\|P_m)$	
Natural Inputs	*Adversarial Examples*	*Natural Inputs*	*Adversarial Examples*
5.14	4.44	**0.82**	**1.07**

Figure 6: We compare the surrogate model with the Mean-field using KL divergence. As the plots show, the surrogate model .

We show in Algorithm 1 how we train the surrogate model along with the mask model when generating adversarial examples. In all our experiments, we work with two-layers deep feedforward neural networks as surrogate models. As stated in the main

paper, the mask model is an hourglass model consisting of convolutional layers. We use a reparameterized Tanh over the outputs of the mask model to define a Bernoulli distribution from which we sample the masks. As the masks consist of binary values, in order to perform backpropagation onto the mask model, we relax the Bernoulli distribution using the Gumbel-Max trick.

Algorithm 1 Generating Adversarial Examples.

Require: Target classifier g_θ
Require: Learning rates α and β
for $(\boldsymbol{x}, \boldsymbol{y}) \sim \boldsymbol{\mathcal{D}}$ **do**
  // This for loop applies only when g_θ is deterministic.
  Draw randomly a mini-batch $\boldsymbol{\mathcal{B}}$ of examples from $\boldsymbol{\mathcal{D}}$
  for $(\boldsymbol{x_c}, \boldsymbol{y_c}) \sim \boldsymbol{\mathcal{B}}$ **do**
    $h_c \leftarrow g_{\bar{\theta}}(\boldsymbol{x_c})$; $\sigma \leftarrow \mathcal{C}_\phi(h_c)$
    Sample $\boldsymbol{z}_1, ..., \boldsymbol{z}_N \sim \mathcal{N}(h_c, \text{diag}(\sigma \odot \sigma))$
    $\phi \leftarrow \phi + \alpha \cdot \nabla_\phi \sum_{i=1}^{N} \log p(\boldsymbol{y_c} | \boldsymbol{z_i}, \phi)$
  Sample a binary mask $\boldsymbol{\omega} \sim p_\nu(\boldsymbol{x})$
  $\boldsymbol{x'} \leftarrow \text{argmax}_{\boldsymbol{x'} \in \mathcal{S}(f(\boldsymbol{x}, \boldsymbol{\omega}))} \ell(g_\theta(\boldsymbol{x'}), \boldsymbol{y})$
  Perform a gradient update on ν using Equation 7 (see main paper)

D.2. Mask Ablation Study

To evaluate the efficacy of our masking mechanism, we do an ablation study. We replace the mask models trained with uncertainty with a random distribution to parameterize a Bernoulli distribution from which we sample the binary masks. We conduct experiments using this ablation against MNIST and CIFAR-10 classifiers and report the results in Table 3. As the results indicate, both *delta* and ReColor perform worse under this setting than under our uncertainty threat model.

Table 3: Model accuracies before and after *delta* and ReColor attacks, under their native threat model and ours compared with the baseline.

	Attacks ↓	*Deterministic Defenses* (%)			*Bayesian Defenses* (%)	
		None	*delta*	*StdAdv*	None	*delta*
MNIST	Vanilla	98.89	98.64	97.33	97.91	76.76
	delta	**0.00**	**92.47**	**0.00**	**0.00**	**0.00**
	Unc. + *delta*	**0.00**	94.79	**0.00**	**0.00**	**0.00**
	Rand. + *delta*	41.16	97.88	6.86	25.87	10.71
CIFAR-10	Vanilla	92.20	84.80	82.88	89.62	77.96
	delta	**0.00**	**30.60**	**0.00**	**0.00**	**1.38**
	Unc. + *delta*	**0.00**	33.32	**0.00**	0.13	3.26
	Rand. + *delta*	**0.00**	61.53	76.51	4.92	29.25
	ReColor	**63.76**	**75.12**	**68.62**	**83.40**	**69.58**
	Unc. + ReColor	65.69	78.64	**68.62**	85.19	71.18
	Rand. + ReColor	83.13	83.18	75.86	86.57	77.65

D.3. Imperceptibility

We evaluate how visually imperceivable the adversarial examples that *delta* and ReColor generate, under our threat model and under their own, based on: *i.*) the sparsity of the binary masks, and *ii.*) by estimating the perceptual distortion in the adversarial examples using LPIPS, a normalized distance measure based on deep network activations [47]. For the former, the results are already summarized in Table 2. For the latter, we compare the adversarial examples with their original inputs using LPIPS on AlexNet.

Table 4: LPIPS Perceptual Distance ($\times 10^{-3}$). **Lower is better.**

	ATTACKS ↓	VANILLA	*delta*
CIFAR-10	*delta*	0.1	9.2
	UNC. + *delta*	**0.0**	**9.1**
	RECOLOR	6.9	11.3
	UNC. + RECOLOR	**1.1**	**10.9**
STL-10	*delta*	7.9	69.0
	UNC. + *delta*	**4.1**	**41.4**
	RECOLOR	22.9	23.3
	UNC. + RECOLOR	**10.3**	**22.6**

LPIPS measures the perceptual distortion between any two images by returning values within $[0, 1]$. We sample at random 100 images from CIFAR-10 and STL-10 and perform *delta* and ReColor attacks under their natural threat models and ours against deterministic classifiers to generate adversarial examples. We compute and report the average LPIPS scores in Table 4. As the results indicate, the adversarial examples retain a greater degree of similarity with the original inputs. To illustrate this further, we provide in Appendix E examples of images and their adversaries.

D.4. Transferability

Adversarial examples often transfer across models [29, 40, 17]. That is, the same examples one generates to evade a specific model can evade other models trained for the same task even when their architectures differ. Here, we ask ourselves: *"How well delta and ReColor attacks transfer under our threat model?"* Explicitly, we seek to validate whether the adversarial examples we generate against deterministic models trained with PGD transfer to stochastic models also defended with PGD, and vice-versa. As Table 5 indicates, both *delta* and ReColor attacks transfer relatively with our threat model in comparison with their own threat model.

Table 5: Transferability of *delta* and ReColor under our model.

	ATTACKS ↓	DETER.	⇆	STOCH.
MNIST	*delta*	54.80		0.53
	UNC. + *delta*	**58.22**		**0.58**
CIFAR-10	*delta*	73.38		9.40
	UNC. + *delta*	**73.60**		**10.04**
	RECOLOR	**66.46**		**38.79**
	UNC. + RECOLOR	63.56		36.19

E. Adversarial Images

Here, we provide more examples of adversarial images that we construct using as natural inputs images drawn from ImageNet and STL-10 datasets.

ImageNet Adversarial Images

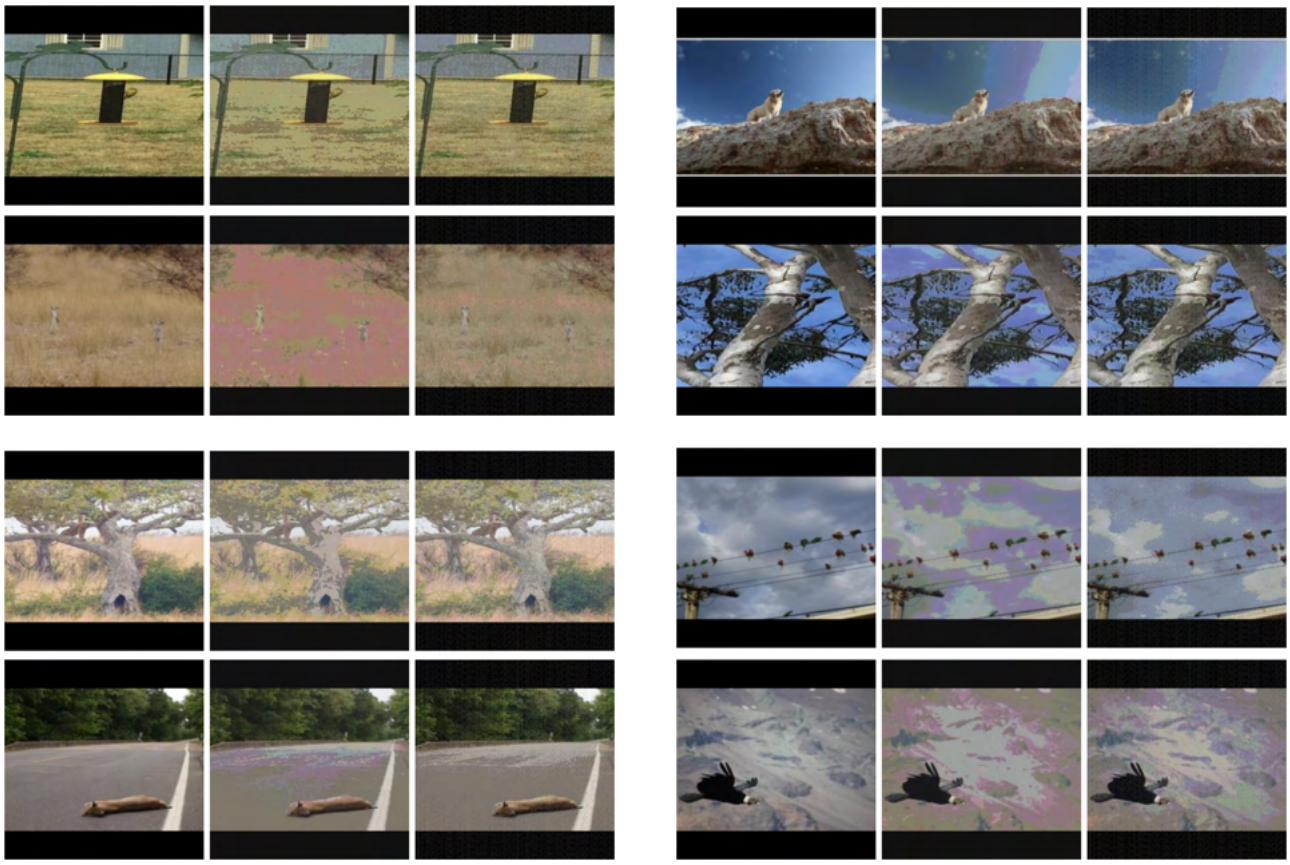


Figure 7: Four groups of ImageNet adversarial examples generated using ReColor under its native functional threat model and under our threat model against an undefended deterministic ResNet-50. From left to right in each row and in each group: original image, adversarial example generated using ReColor, and adversarial example generated using ReColor under our threat model.

Continued next page ...

STL-10 Adversarial Images

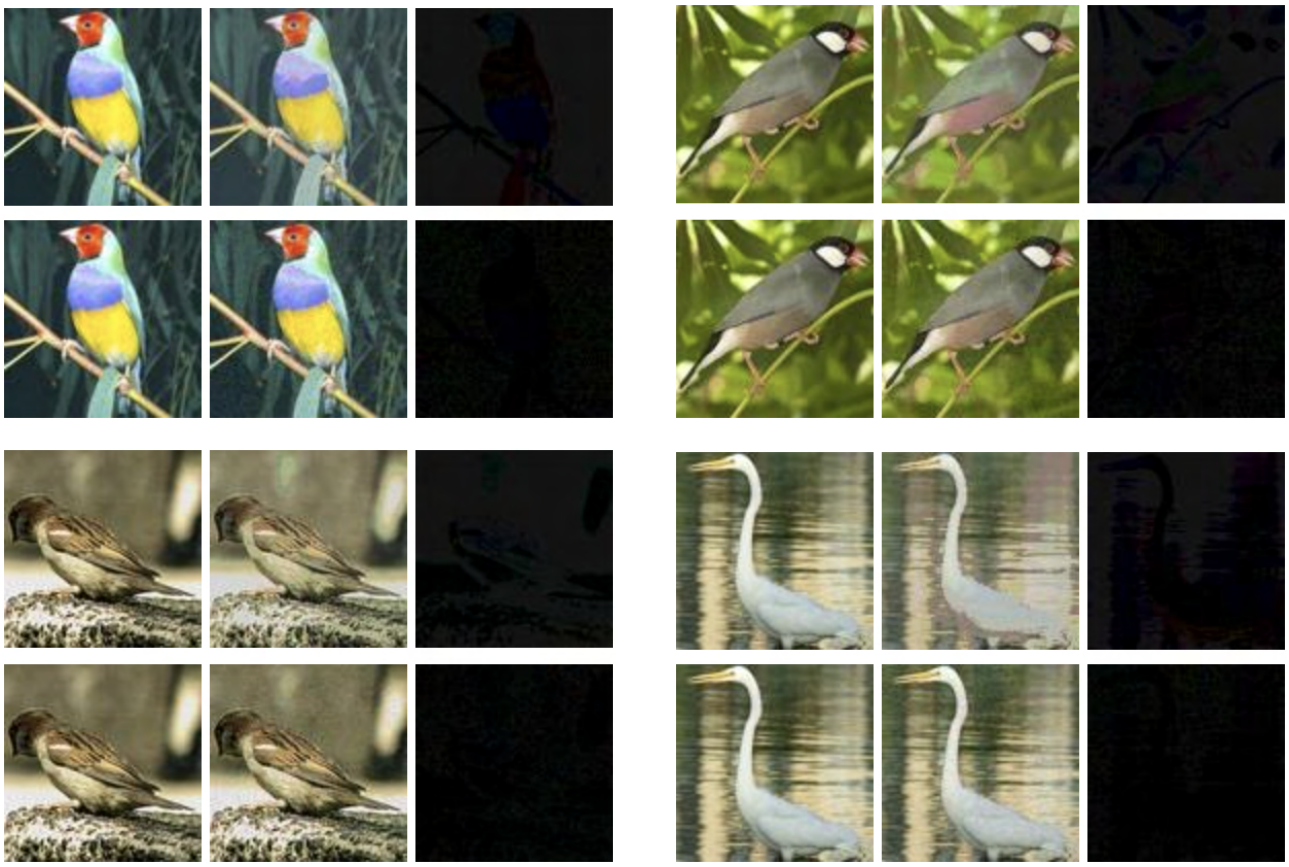


Figure 8: Four groups of STL-10 adversarial examples generated using ReColor under its native functional threat model and under our threat model against an undefended deterministic classifier. From left to right in each group: *top row* — original image, adversarial example generated using ReColor, and perturbations, *bottom row* — original image, adversarial example generated using ReColor under our threat model, and perturbations.