Adversarial Examples Are Not Bugs, They Are Features

Ilyas, Andrew, et al. MIT

Sein Jang

tpdls24@gmail.com

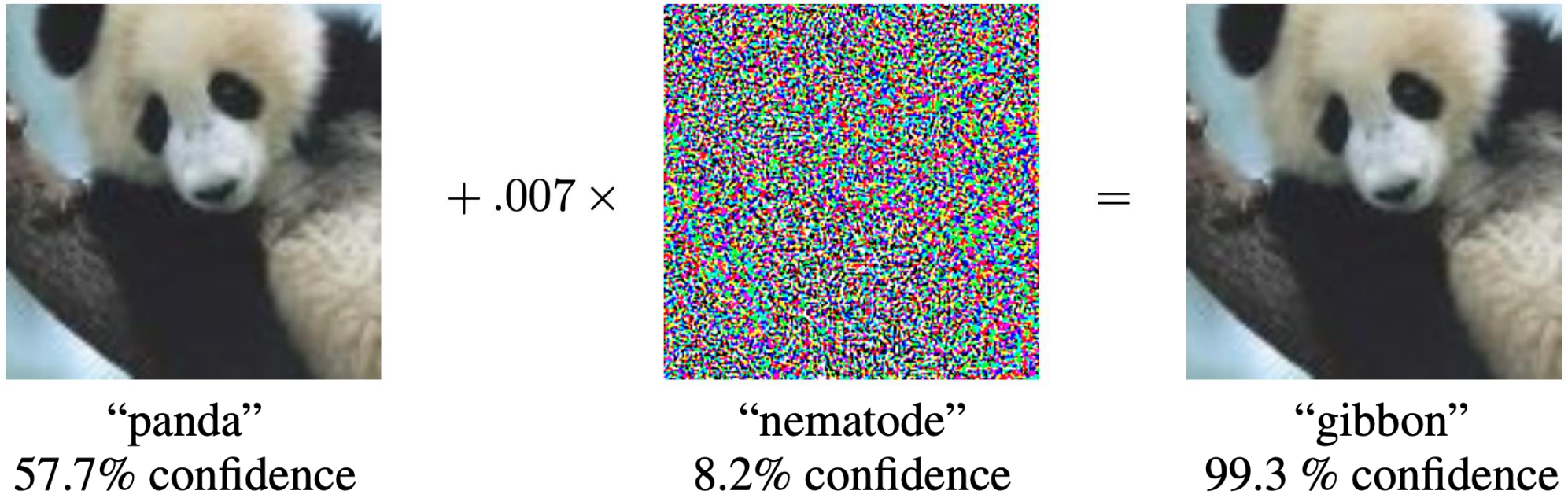
June 1, 2020

Contents

- Adversarial Examples?
- Useful, Robust, and Non-Robust Features
- Finding Robust Features
- Studying (Non)-Robust Features
- Conclusion

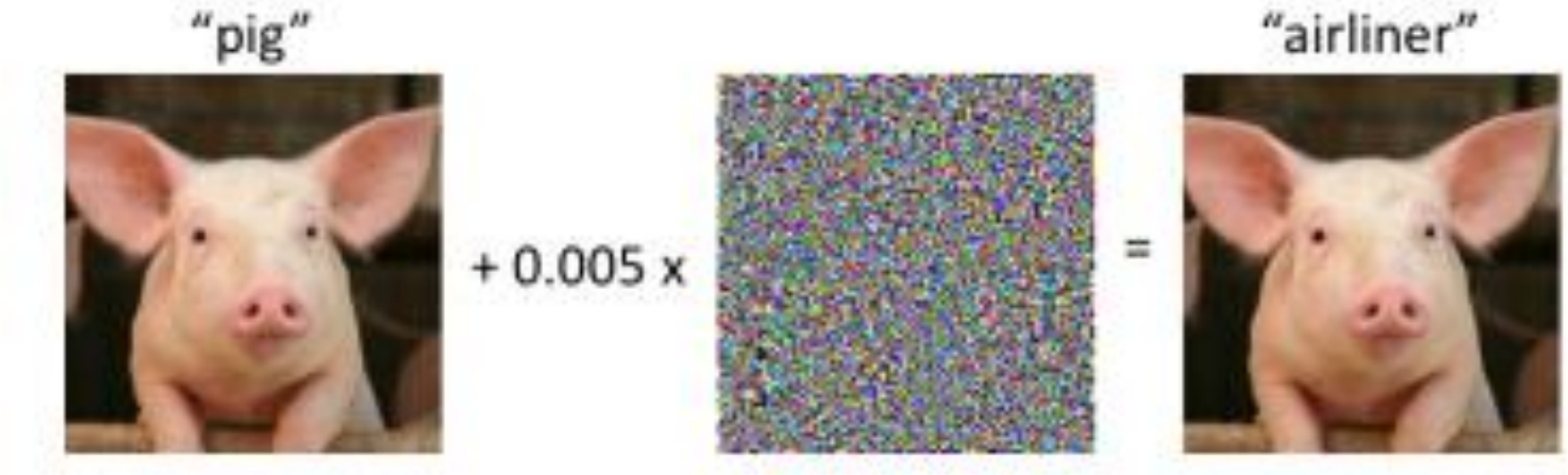
Adversarial Attack

An **adversarial attack** consists of subtly **modifying an original image** in such a way that the **changes are almost undetectable to the human eye**. The modified image is called an adversarial image, and when submitted to a classifier is misclassified, while the original one is correctly classified.

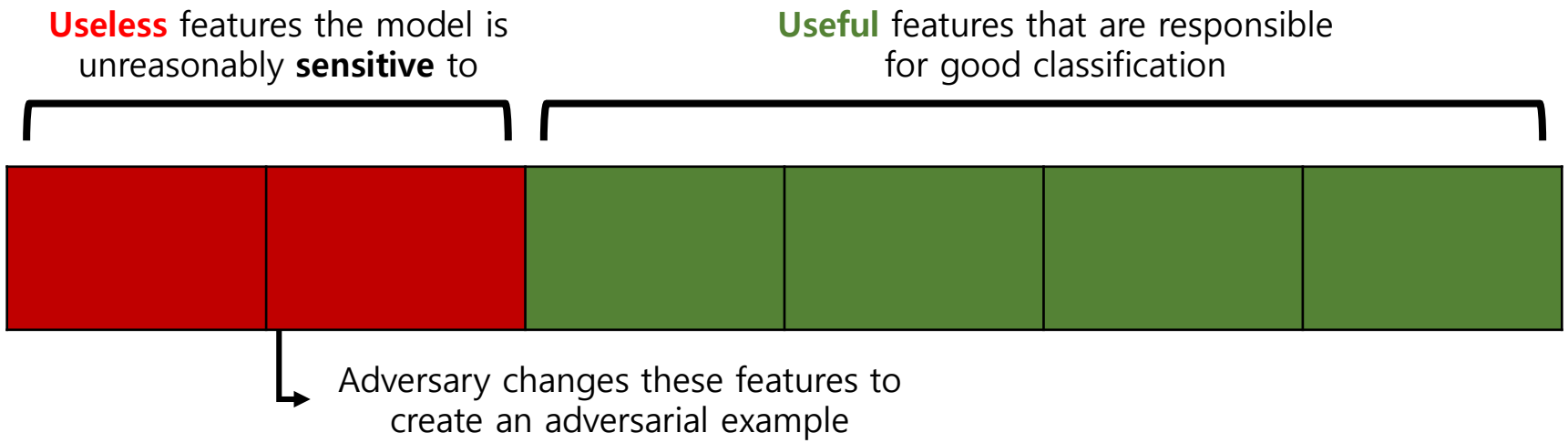


Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples"

Adversarial Examples?



Why are ML models so sensitive to small perturbations?



Bug? Features?

Previous work has proposed a variety of explanations for Adversarial Example

- Theoretical models

Ludwig Schmidt et al. "Adversarially Robust Generalization Requires More Data". In: Advances in Neural Information Processing Systems (NeurIPS). 2018.

Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. "Adversarial examples from computational constraints". In: arXiv preprint arXiv:1805.10204. 2018.

- High-dimensions nature of the input space or statistical fluctuations in the training data

Justin Gilmer et al. "Adversarial spheres". In: Workshop of International Conference on Learning Representations (ICLR). 2018.

Ali Shafahi et al. "Are adversarial examples inevitable?" In: International Conference on Learning Representations (ICLR). 2019.

But,

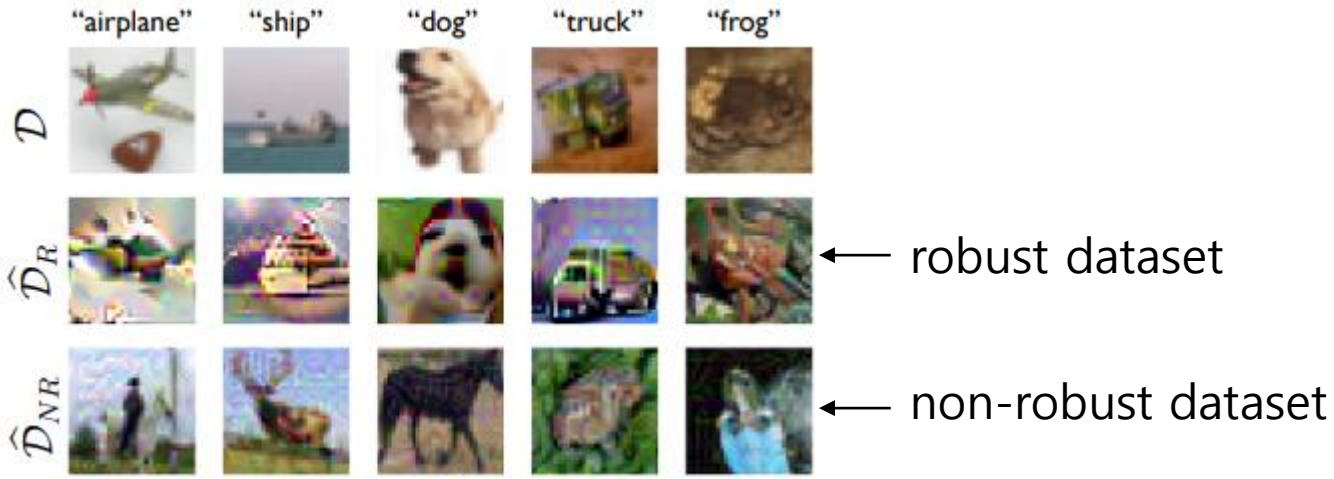
In this work, we propose a new perspective on the phenomenon of adversarial examples. In contrast to the previous models, we cast adversarial vulnerability as a fundamental consequence of the dominant supervised learning paradigm. Specifically, we claim that:

Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data.

Bug? Features?

How we usually train classifiers?

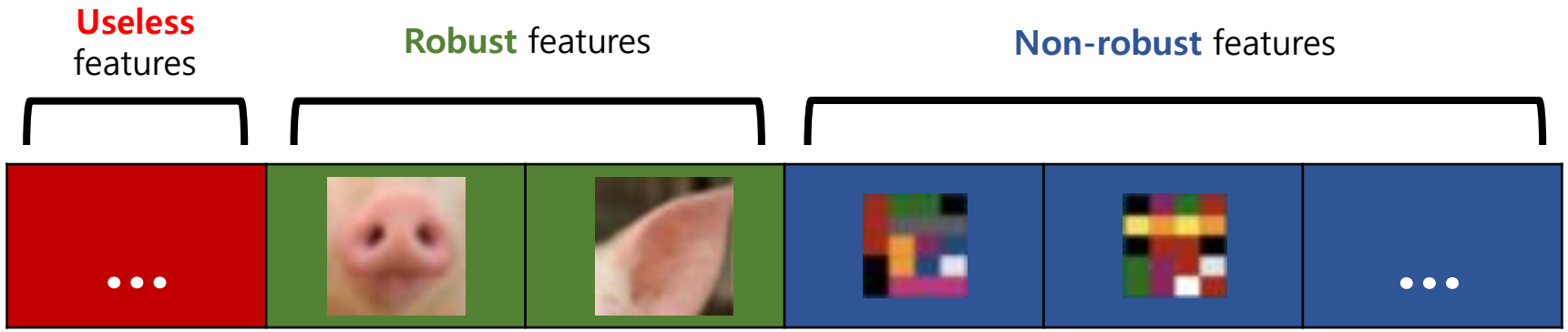
- we usually train classifiers to maximize accuracy
- classifiers tend to use any available signal to do so, even those that look incomprehensible to humans
- "a tail" or "ears" is no more natural to a classifier than any other equally predictive feature
- *standard ML datasets do admit highly predictive yet imperceptible features* ***(non-robust)***



Bug? Features?

The dataset contains

- Useless, Robust, Non-robust features



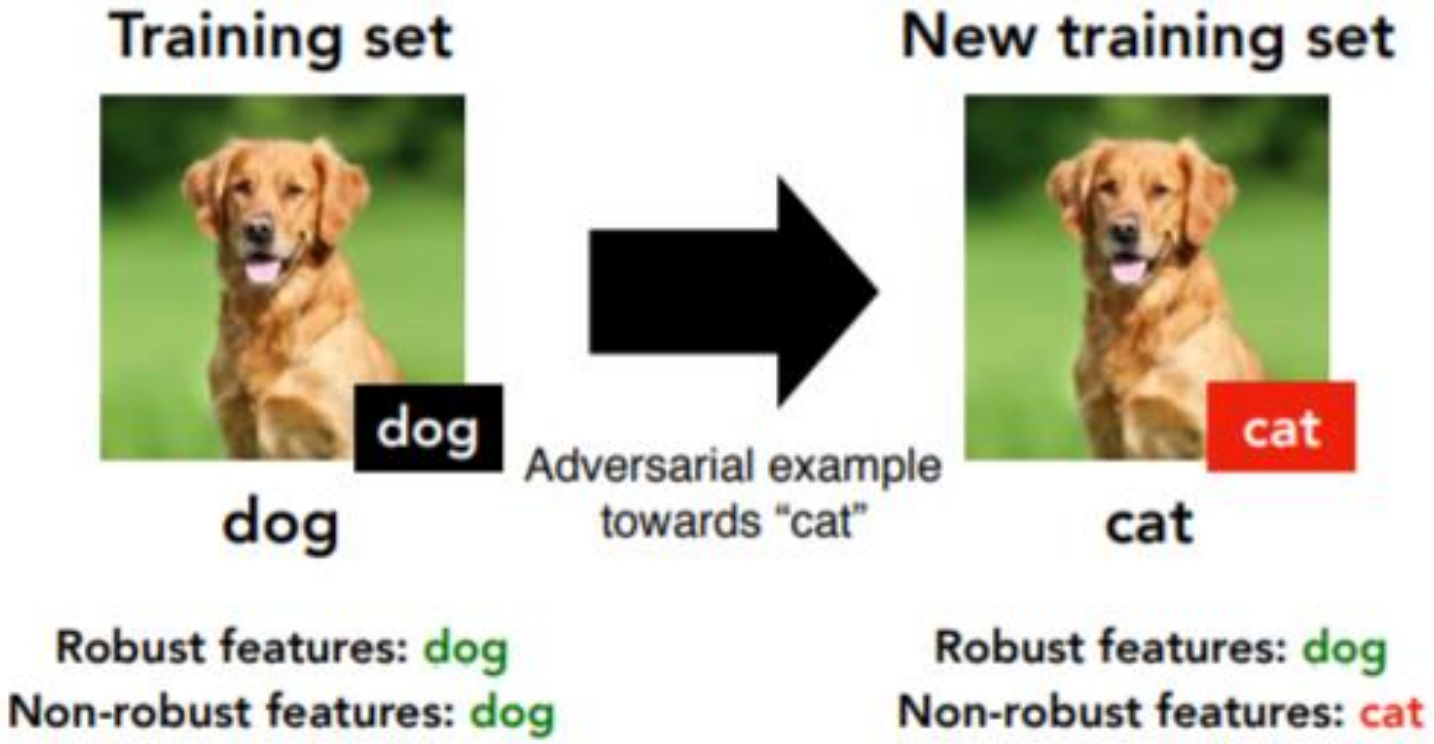
To train model for maximize the accuracy, models want to use non-robust features.

Because the non-robust features are often good …

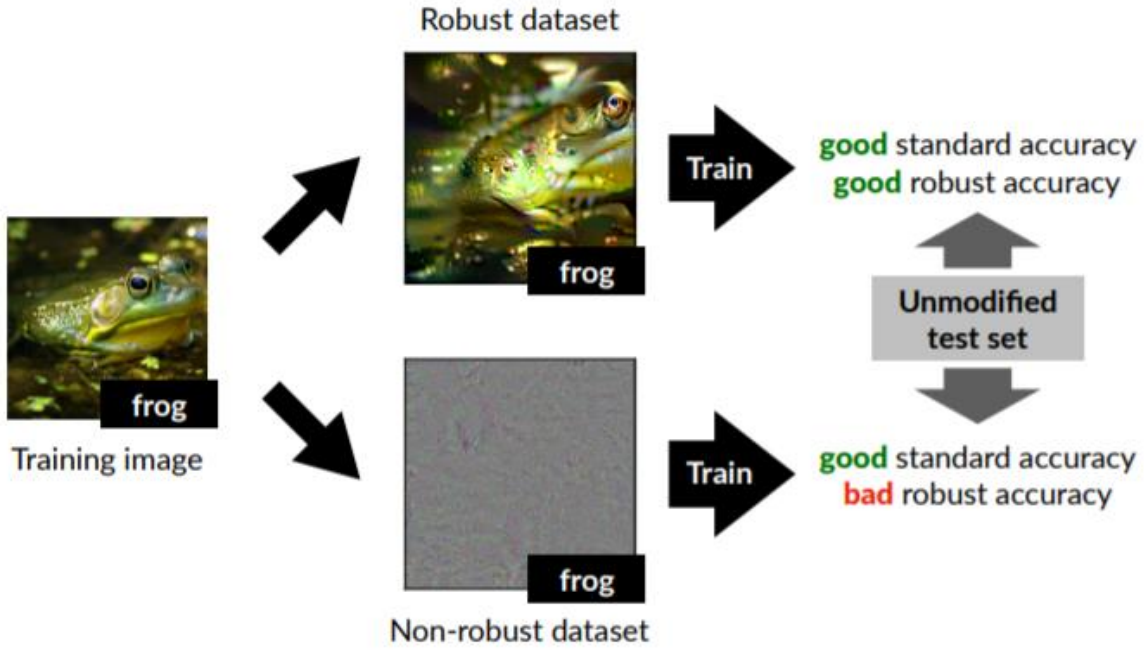
Bug? Features?

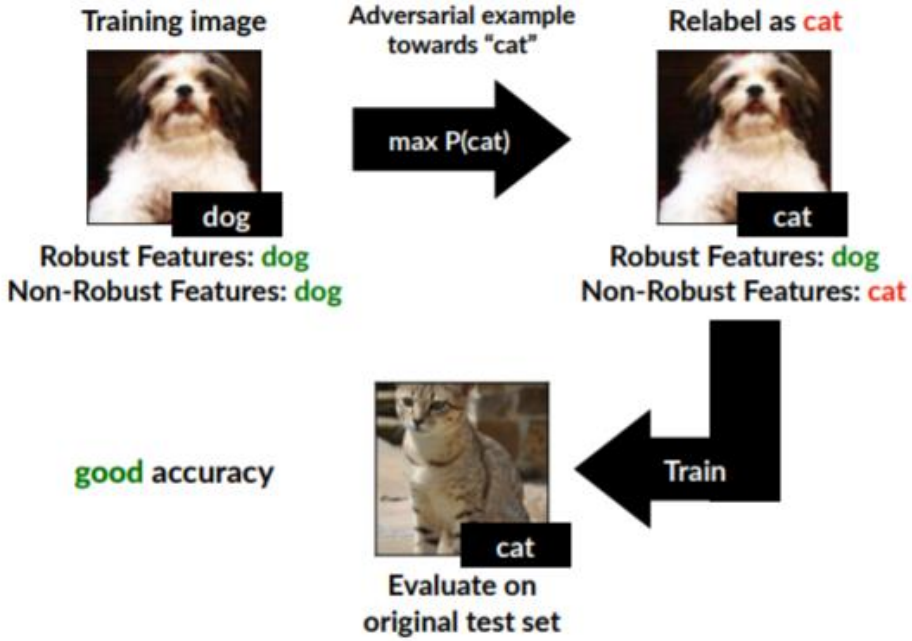
Adversarial Examples are …

- If the models use non-robust features -> adversarial examples



The Robust Features Model





Construct two types classification

- A "robustified" version for robust classification (Figure (a))
- A "non-robust" version for standard classification (Figure (b))

The Robust Features Model

Setup. We consider binary classification[3], where input-label pairs $(x, y) \in \mathcal{X} \times \{\pm 1\}$ are sampled from a (data) distribution $\mathcal{D}$; the goal is to learn a classifier $C : \mathcal{X} \to \{\pm 1\}$ which predicts a label y corresponding to a given input x.

We define a *feature* to be a function mapping from the input space $\mathcal{X}$ to the real numbers, with the set of all features thus being $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$. For convenience, we assume that the features in $\mathcal{F}$ are shifted/scaled to be mean-zero and unit-variance (i.e., so that $\mathbb{E}_{(x,y)\sim\mathcal{D}}[f(x)] = 0$ and $\mathbb{E}_{(x,y)\sim\mathcal{D}}[f(x)^2] = 1$), in order to make the following definitions scale-invariant[4]. Note that this formal definition also captures what we abstractly think of as features (e.g., we can construct an f that captures how "furry" an image is).

Useful, Robust, and Non-Robust Features

ρ-useful features: For a given distribution $\mathcal{D}$, we call a feature f *ρ-useful* ($\rho > 0$) if it is correlated with the true label in expectation, that is if

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[y \cdot f(x)] \geq \rho. \tag{1}$$

We then define $\rho_{\mathcal{D}}(f)$ as the largest ρ for which feature f is ρ-useful under distribution $\mathcal{D}$. (Note that if a feature f is negatively correlated with the label, then $-f$ is useful instead.) Crucially, a linear classifier trained on ρ-useful features can attain non-trivial generalization performance.

γ-robustly useful features: Suppose we have a ρ-useful feature f ($\rho_{\mathcal{D}}(f) > 0$). We refer to f as a *robust feature* (formally a γ-robustly useful feature for $\gamma > 0$) if, under adversarial perturbation (for some specified set of valid perturbations Δ), f remains γ-useful. Formally, if we have that

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\inf_{\delta\in\Delta(x)} y \cdot f(x+\delta)\right] \geq \gamma. \tag{2}$$

Useful, non-robust features: A *useful, non-robust feature* is a feature which is ρ-useful for some ρ bounded away from zero, but is not a γ-robust feature for any $\gamma \geq 0$. These features help with classification in the standard setting, but may hinder accuracy in the adversarial setting, as the correlation with the label can be flipped.

Standard Training vs Robust Training

Standard Training. Training a classifier is performed by minimizing a loss function (via *empirical risk minimization* (ERM)) that decreases with the correlation between the weighted combination of the features and the label. The simplest example of such a loss is [5]

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}_\theta(x,y)\right] = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y\cdot\left(b+\sum_{f\in F} w_f\cdot f(x)\right)\right]. \tag{3}$$

When minimizing classification loss, *no distinction* exists between robust and non-robust features: the only distinguishing factor of a feature is its ρ-usefulness. Furthermore, the classifier will utilize *any* ρ-useful feature in F to decrease the loss of the classifier.

Robust training. In the presence of an *adversary*, any useful but non-robust features can be made *anti-correlated* with the true label, leading to adversarial vulnerability. Therefore, ERM is no longer sufficient to train classifiers that are robust, and we need to explicitly account for the effect of the adversary on the classifier. To do so, we use an *adversarial* loss function that can discern between robust and non-robust features [Mad+18]:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\Delta(x)} \mathcal{L}_\theta(x+\delta,y)\right], \tag{4}$$

for an appropriately defined set of perturbations Δ. Since the adversary can exploit non-robust features to degrade classification accuracy, minimizing this adversarial loss (as in adversarial training [GSS15; Mad+18]) can be viewed as explicitly preventing the classifier from learning a useful but non-robust combination of features.

Finding Robust (and Non-Robust) Features

Disentangling robust and non-robust features

Only **robust features** are useful → Standard Training → **Robust** classifier

We cannot directly manipulate the features of very complex, high-dimensional datasets (X)

We will leverage a robust model and modify our dataset to contain only the features that are relevant to that model (O)

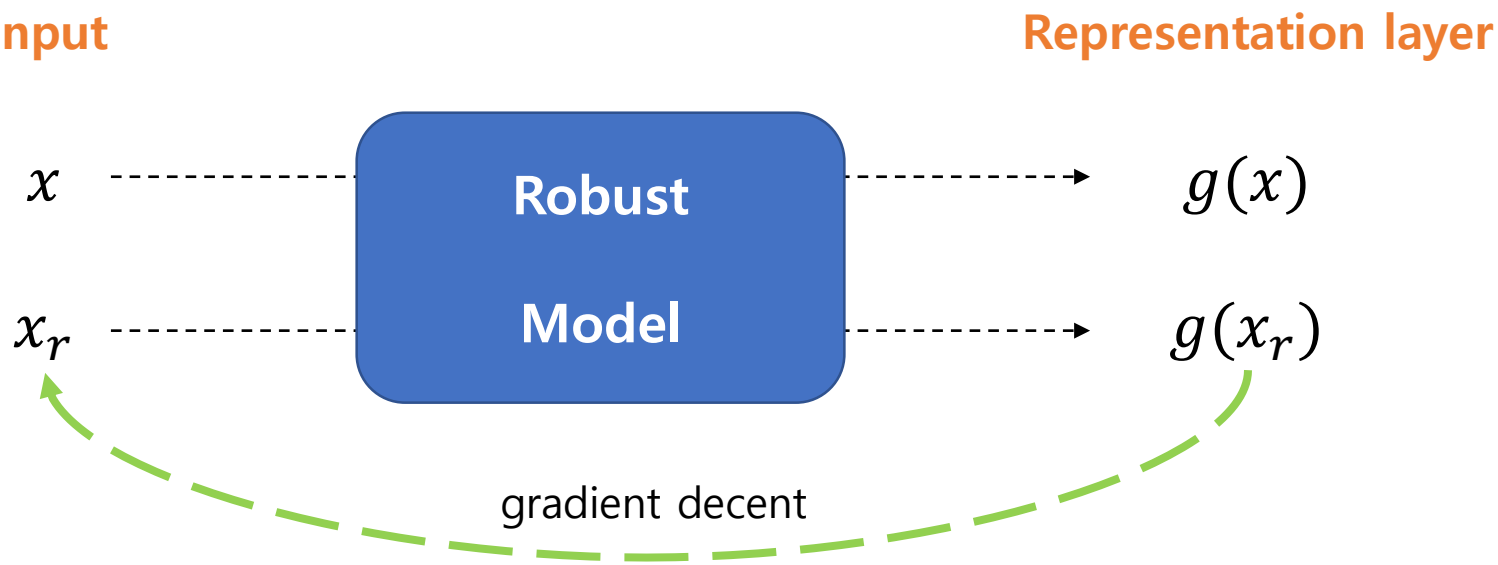
$$\mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}_R}\left[f(x)\cdot y\right] = \begin{cases} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[f(x)\cdot y\right] & \text{if } f \in F_C \\ 0 & \text{otherwise,} \end{cases}$$

Set of features utilized by C

Robust model

Finding Robust (and Non-Robust) Features

Disentangling robust and non-robust features



$$\min_{x_r} \|g(x_r) - g(x)\|_2,$$

Finding Robust (and Non-Robust) Features

Constructing a Robust Dataset

GETROBUSTDATASET(D)

1. $C_R \leftarrow$ ADVERSARIALTRAINING(D)
   $g_R \leftarrow$ mapping learned by C_R from the input to the representation layer
2. $D_R \leftarrow \{\}$
3. For $(x, y) \in D$
   $x' \sim D$
   $x_R \leftarrow \arg\min_{z \in [0,1]^d} \|g_R(z) - g_R(x)\|_2$ # Solved using ℓ_2-PGD starting from x'
   $D_R \leftarrow D_R \bigcup \{(x_R, y)\}$
4. Return D_R

Finding Robust (and Non-Robust) Features

Standard and Robust accuracy on the CIFAR-10 test set

- Classifier learned using the **robust dataset** attains **good accuracy in both standard and adversarial** settings
- Classifier learned using the **non-robust dataset** leads to **good standard accuracy**, yet yields almost **no robustness**

Non-robust features suffice for standard classification

Construct a dataset where the only features that are useful for classification are non-robust features

Modify Input – label pair

$$(\boldsymbol{x}, \boldsymbol{y}) \rightarrow (\boldsymbol{x_{adv}}, \boldsymbol{t})$$

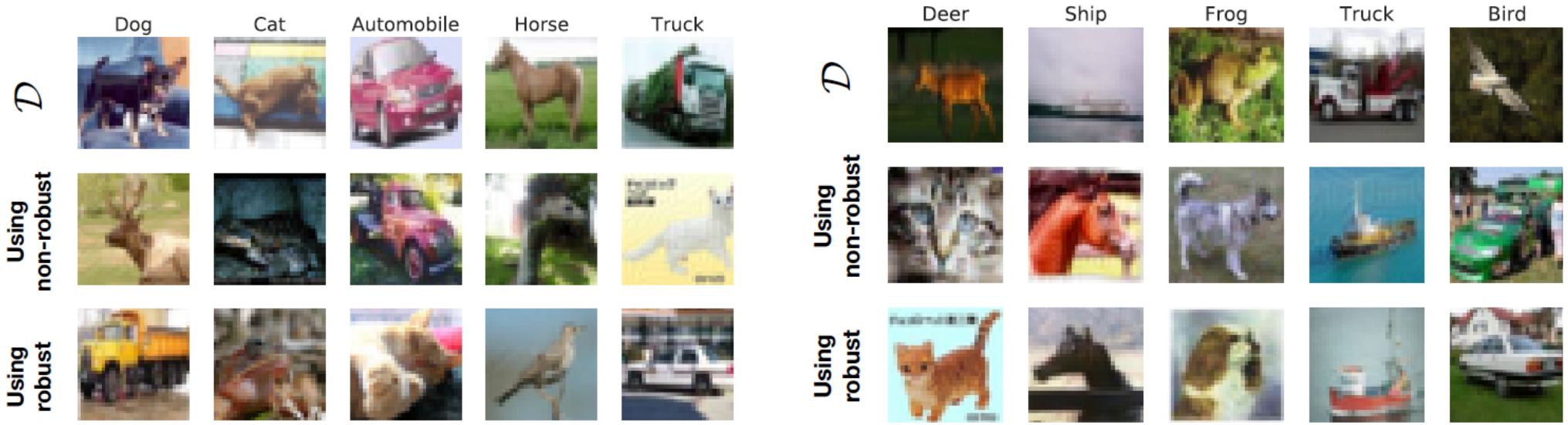
$$x_{adv} = \underset{\|x'-x\| \leq \varepsilon}{\arg\min} \; L_C(x', t),$$

Class t either,

(a) Uniformly at **random** among classes

(b) **Deterministically** according to the source class

Non-robust features suffice for standard classification



(a) $\widehat{\mathcal{D}}_{rand}$ (b) $\widehat{\mathcal{D}}_{det}$

Figure 9: Random samples from datasets where the input-label correlation is entirely based on non-robust features. Samples are generated by performing small adversarial perturbations using either random ($\widehat{\mathcal{D}}_{rand}$) or deterministic ($\widehat{\mathcal{D}}_{det}$) label-target mappings for every sample in the training set. Each image shows: *top*: original; *middle*: adversarial perturbations using a standard ERM-trained classifier; *bottom*: adversarial perturbations using a robust classifier (adversarially trained against $\varepsilon = 0.5$).

Non-robust features suffice for standard classification

Non-robust features are indeed useful for classification on the standard setting.

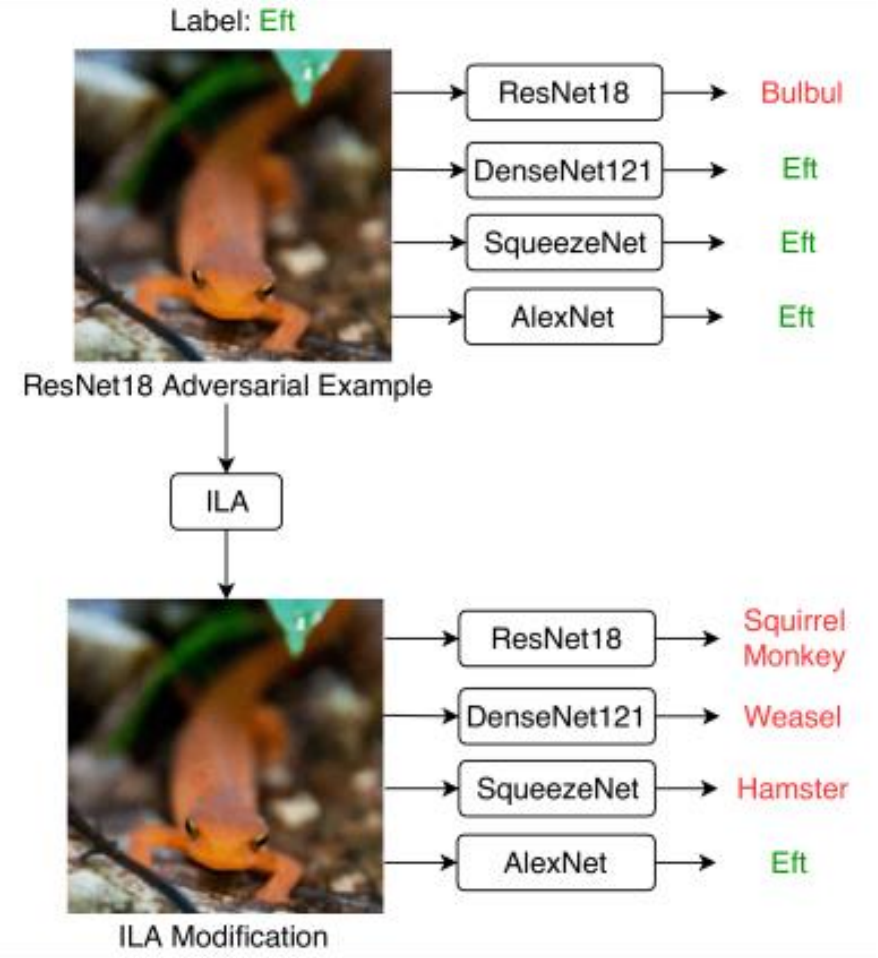
Remarkably, even training on $\widehat{D}_{det}$(where all the robust features are correlated with the wrong class), results in a well-generalizing classifier.

This indicates that **non-robust features can be picked up by models during standard training, even in the presence of robust features that are predictive**

Source Dataset	Dataset	
	CIFAR-10	ImageNet_R
$\mathcal{D}$	95.3%	96.6%
$\widehat{\mathcal{D}}_{rand}$	63.3%	87.9%
$\widehat{\mathcal{D}}_{det}$	43.7%	64.4%

Adversarial Example Transferability

Adversarial examples are often **transfer** : The same adversarial example fools more than one model

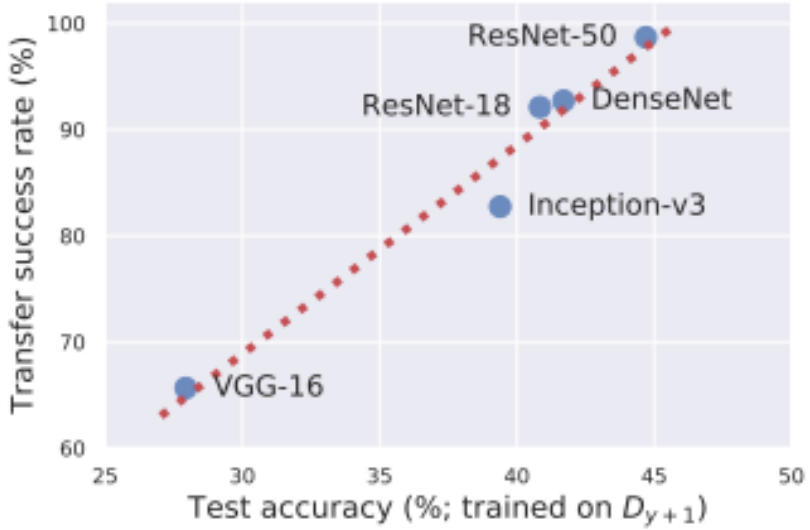


Huang, Qian, et al. "Enhancing adversarial example transferability with an intermediate level attack." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

Adversarial Example Transferability

Transferability can arise from non-robust features

- Adversarial examples can arise as a result of perturbing well-generalizing, yet brittle features.
- Different classifiers trained on independent samples from that distribution are likely to utilize similar non-robust features.
- Consequently, an adversarial example constructed by exploiting the **non-robust features learned by one classifier will transfer to any other classifier utilizing these features in a similar** manner.



Conclusion

The phenomenon of adversarial examples is a natural consequence of the presence of ***highly predictive but non-robust features*** in standard ML datasets.

From the perspective of interpretability, as long as models rely on these non-robust features, we cannot expect to have model explanations that are both human-meaningful and faithful to the models themselves.

Overall, attaining models that are robust and interpretable will require explicitly encoding ***human priors*** into the training process.

Q & A