Evasion Attacks

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Trustworthy and Secure AI Lab

October 31, 2025

Today's Agenda

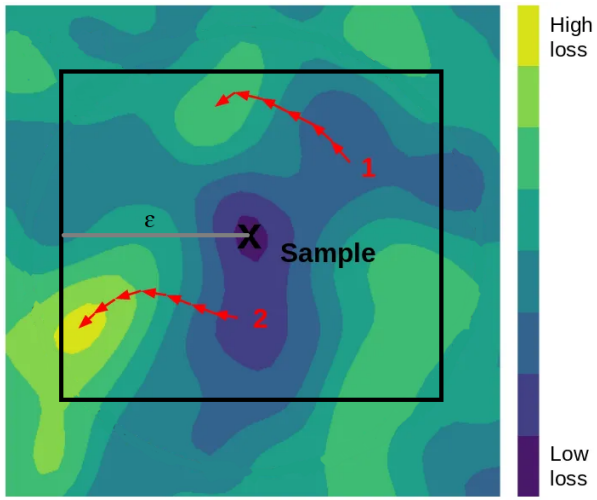
Projected Gradient Descent

Projected Gradient Descent (PGD) Attack

A more powerful adversary is the **multi-step variant**, which is essentially projected gradient descent (PGD) on the negative loss function:

$$
\begin{aligned}
x^{(0)} &= \text{clip}_{[0,1]}\{x + \mathcal{U}(-\epsilon, +\epsilon)\}, \\
\delta^{(t+1)} &= \alpha \cdot \text{sign}\left(\nabla_x \mathcal{L}(f_\theta(x^{(t)}), y)\right), \\
x^{(t+1)} &= \text{clip}_{[\max(0,x-\epsilon),\min(1,x+\epsilon)]}\{x^{(t)} + \delta^{(t+1)}\}
\end{aligned}
$$

where x is a natural data, $\mathcal{U}$ is uniform distribution, $\text{clip}_{[a,b]}\{x\}$ function is used to trim values outside interval $[a, b]$ to the interval edges, ϵ is the radius of allowed perturbation $\|\cdot\|_\infty \leq \epsilon$, and t is iteration index.



Source

PGD Algorithm

Algorithm 1 Projected gradient descent.

Input: clean data x, true labels y, model f
Parameters: perturbation budget ϵ, step size α, random initialization `Random Noise`, number of steps k
if `Random Noise` **then**
    $x^{(0)} \leftarrow x + \mathcal{U}(-\epsilon, +\epsilon)$
    $x^{(0)} \leftarrow \text{clip}_{[0,1]}\{x^{(0)}\}$ ▷ Ensure valid pixel range
else
    $x^{(0)} \leftarrow x$
end if
for $t = 1, \ldots, k$ **do**
    $\delta^{(t)} \leftarrow \text{sign}\left(\nabla \mathcal{L}\left(f(x^{(t-1)}, y)\right)\right)$
    $x^{(t)} \leftarrow x^{(t-1)} + \alpha \cdot \delta^{(t)}$
    $x^{(t)} \leftarrow \text{clip}_{[x-\epsilon, x+\epsilon]}\{x^{(t)}\}$ ▷ Enforce perturbation budget
    $x^{(t)} \leftarrow \text{clip}_{[0,1]}\{x^{(t)}\}$ ▷ Ensure valid pixel range
end for

The Landscape of Adversarial Examples

- While there are many local maxima spread widely apart within $x_i + \mathcal{S}$, they tend to have **very well-concentrated loss values**.
- This echoes the folklore belief that training neural networks is possible because the loss (as a function of model parameters) typically has many local minima with very similar values.

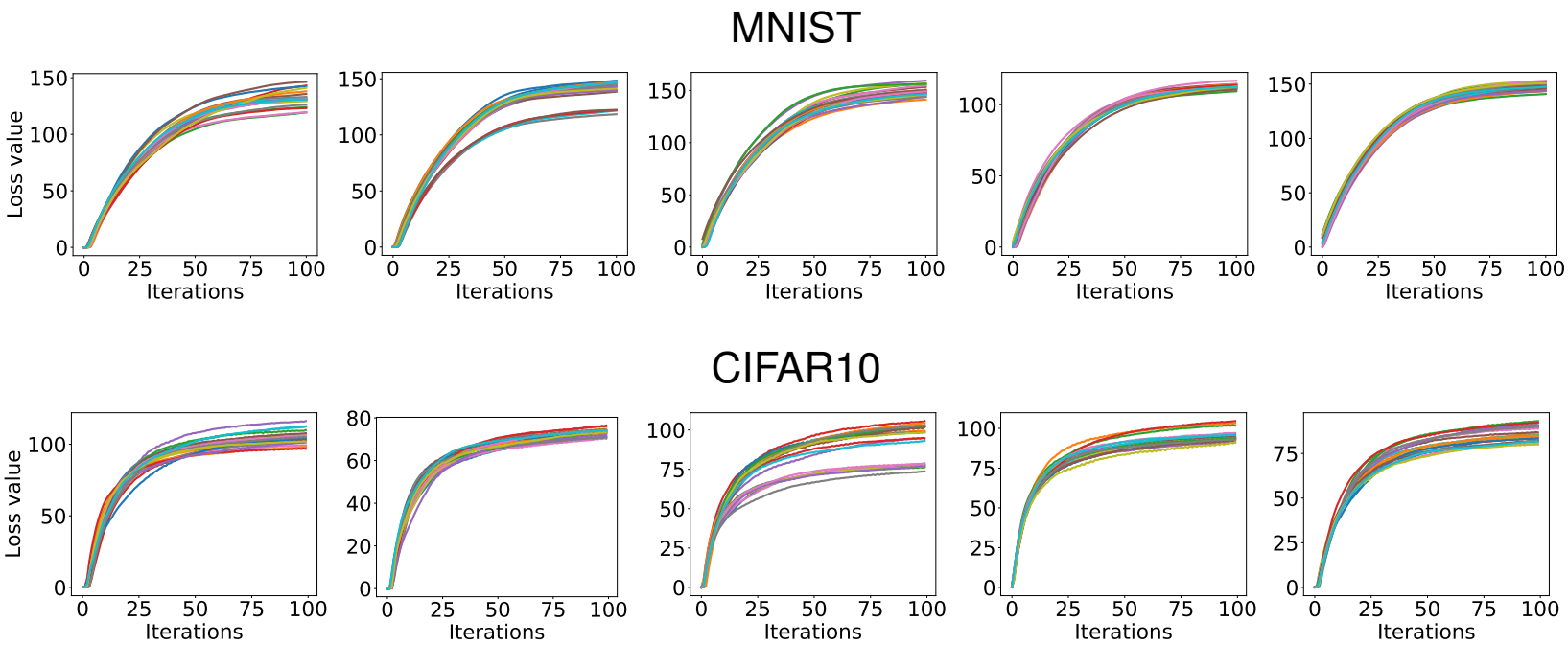


Figure: Loss function value over PGD iterations for 20 random restarts on random examples.

First-Order Adversaries

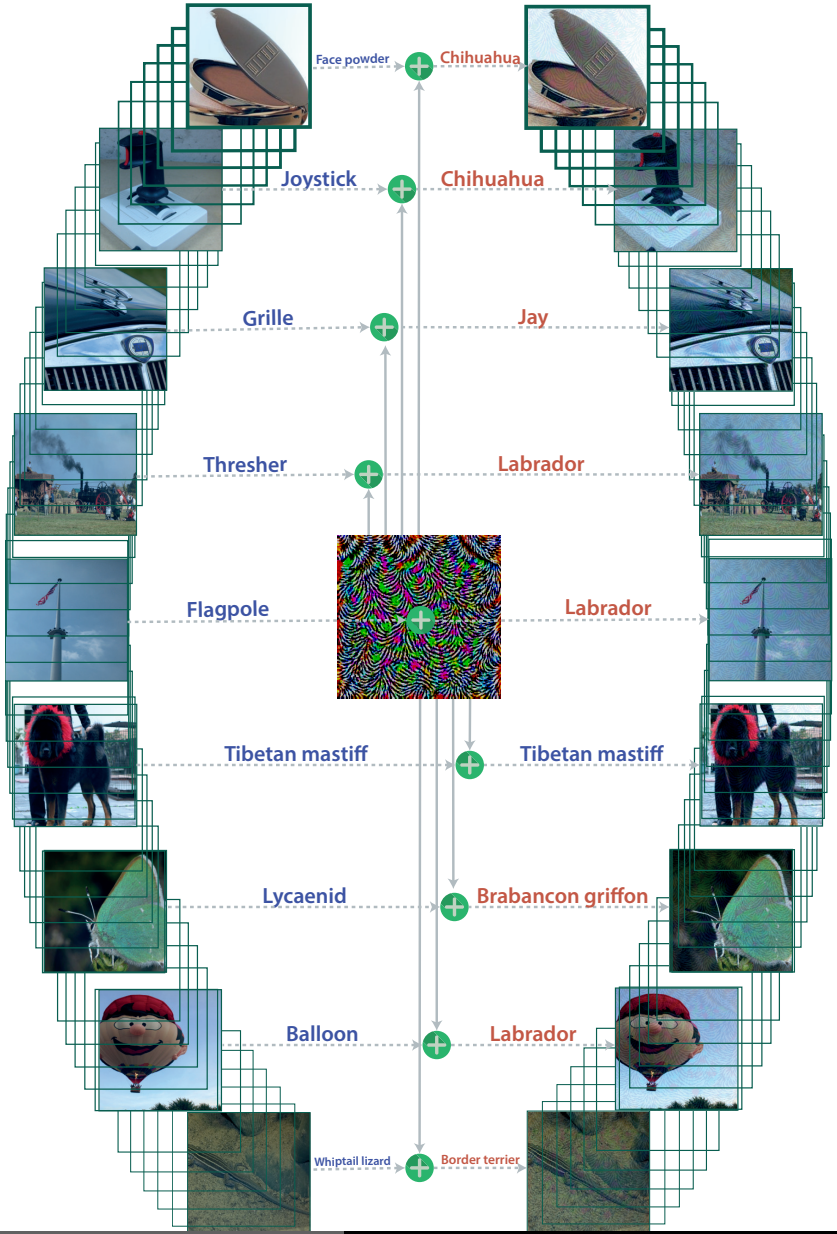
The concentration phenomenon suggests an intriguing view on the problem in which robustness against the PGD adversary yields robustness against all first-order adversaries:

- As long as the adversary only uses gradients of the loss function with respect to the input, we conjecture that it will not find significantly better local maxima than PGD.
- Of course, our exploration with PGD does not preclude the existence of some isolated maxima with much larger function value.
- Experiments suggest that such better local maxima are hard to find with first order methods.

Universal Adversarial Perturbations

Universal (Image-agnostic) Perturbations

- We show the existence of a **universal (image-agnostic)** and very small perturbation vector that causes natural images to be misclassified with high probability.
  - They generalize very well across neural networks.
- The surprising existence of universal perturbations reveals important **geometric correlations among the high-dimensional decision boundary** of classifiers.



Universal Perturbations

Let $\mathcal{D}$ denote a distribution of images in $\mathbb{R}^d$, and $\hat{k}$ define a classification function that outputs for each image $x \in \mathbb{R}^d$ an estimated label $\hat{k}(x)$. Let $X = \{x_1, \ldots, x_m\}$ be a set of images sampled from the distribution $\mathcal{D}$.

The main focus of this paper is to seek perturbation vectors $v \in \mathbb{R}^d$ that fool the classifier $\hat{k}$ on almost all data points sampled from $\mathcal{D}$. That is, we seek a vector v such that

$$\hat{k}(x + v) \neq \hat{k}(x) \text{ for } \textbf{most } x \sim \mathcal{D}$$

The goal is to find v that satisfies the following two constraints:

1. $\|v\|_p \leq \xi$
2. $\underset{x \sim \mathcal{D}}{\mathbb{P}} \left(\hat{k}(x + v) \neq \hat{k}(x) \right) \geq 1 - \delta$

The parameter ξ controls the magnitude of the perturbation vector v, and δ quantifies the desired fooling rate for all images sampled from the distribution $\mathcal{D}$.

UAP Algorithm

- Algorithm 2 proceeds iteratively over the data points in X and gradually builds the universal perturbation.
- At each iteration,
  - If the current universal perturbation v does not fool $\hat{k}$ for data point x_i, we seek the extra perturbation Δv_i with minimal norm that allows to fool $\hat{k}$ for data point x_i by solving the following optimization problem:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \quad \text{s.t.} \quad \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

  - To ensure that the constraint $\|v\|_p \leq \xi$ is satisfied, the updated universal perturbation is further projected on the ℓ_p ball of radius ξ and centered at 0. That is, let $\mathcal{P}_{p,\xi}$ be the projection operator defined as follows:

$$\mathcal{P}^{(v)}_{p,\xi}(x) = \arg\min_{v'} \|v' - x\|_2 \quad \text{subject to } \|v' - v\|_p \leq \xi.$$

  - Then, our update rule is given by $v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i)$.
  - if p = 2, then $\mathcal{P}^{(v)}_{2,\epsilon}(v + \Delta v) = v + \min\left(1, \frac{\epsilon}{\|\Delta v\|_2}\right) \Delta v$.
- Several passes on the data set X are performed to improve the quality of the universal perturbation.
- The algorithm is terminated when the empirical "fooling rate" on the perturbed data set $X_v := \{x_1 + v, \ldots, x_m + v\}$ exceeds the target threshold $1 - \delta$. That is, we stop the algorithm whenever:

$$\text{Err}(X_v) := \frac{1}{m} \sum_{i=1}^{m} 1_{\hat{k}(x_i+v) \neq \hat{k}(x_i)} \geq 1 - \delta.$$

UAP Algorithm

Algorithm 2 Computation of universal perturbations.

Inputs: Data points X, classifier $\hat{k}$, desired ℓ_p norm of the perturbation ξ, desired accuracy on perturbed samples δ.
Outputs: Universal perturbation vector v.
Initialize $v \leftarrow 0$.
while $\text{Err}(X_v) \leq 1 - \delta$ **do**
    for each datapoint $x_i \in X$ **do**
        if $\hat{k}(x_i + v) = \hat{k}(x_i)$ **then**
            Compute the *minimal* perturbation that sends $x_i + v$ to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

            Update the perturbation:

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$

        end if
    end for
end while

UAP Algorithm Properties

- Universal perturbation is **class-agnostic**
- Interestingly, in practice, the number of data points $|X| = m$ need not be large to compute a universal perturbation.
  - we can set m to be much smaller than the number of training points.
- L-BFGS, FGSM, PGD, or DeepFool attacks can be used to compute Δv_i.
- Different random shuffling of the set X naturally lead to **a diverse set of universal perturbations** v satisfying the required constraints.

Fooling Ratios

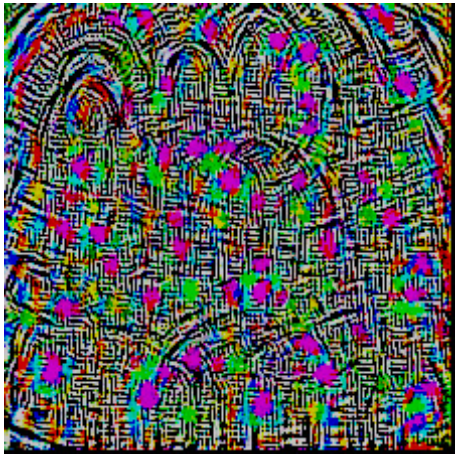
The table reports the fooling ratio, that is the proportion of images that change labels when perturbed by universal perturbation.

- Results are reported for ℓ_2 and ℓ_∞ norms, where we respectively set $\xi = 2000$ and $\xi = 10/255$.
- Each result is reported on the set X, which is used to compute the perturbation, as well as on the 50,000 **validation** images.

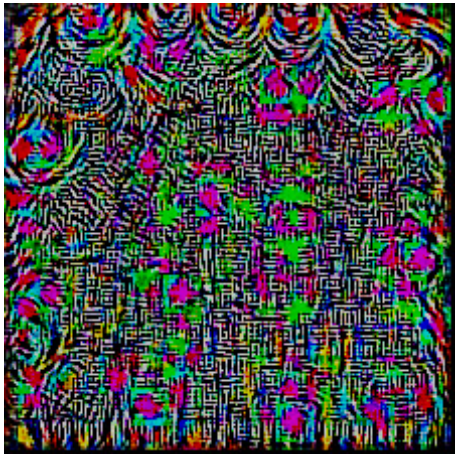
Norm	Data	CaffeNet	VGG-F	VGG-16	VGG-19	GoogLeNet	ResNet-152
ℓ_2	X	85.4%	85.9%	90.7%	86.9%	82.9%	89.7%
	Val.	85.6%	87.0%	90.3%	84.5%	82.0%	88.5%
ℓ_∞	X	93.1%	93.8%	78.5%	77.8%	80.8%	85.4%
	Val.	93.3%	93.7%	78.3%	77.8%	78.9%	84.0%

- These results have an element of surprise, as they show the existence of **single universal perturbation vectors that cause natural images to be misclassified with high probability**, albeit being quasi-imperceptible to humans.

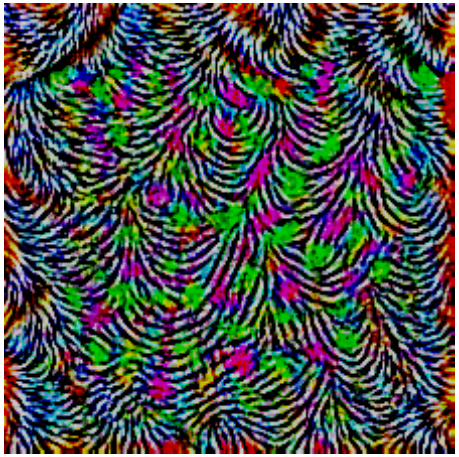
Examples of Universal Perturbations



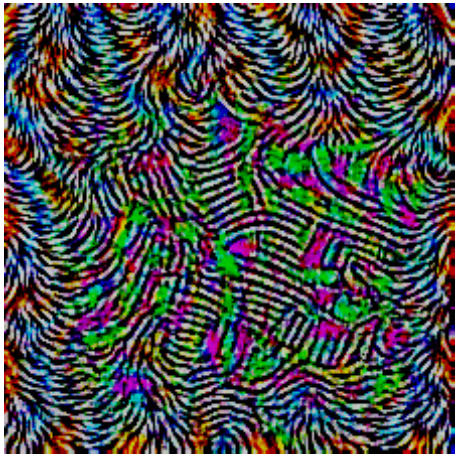
(a) CaffeNet



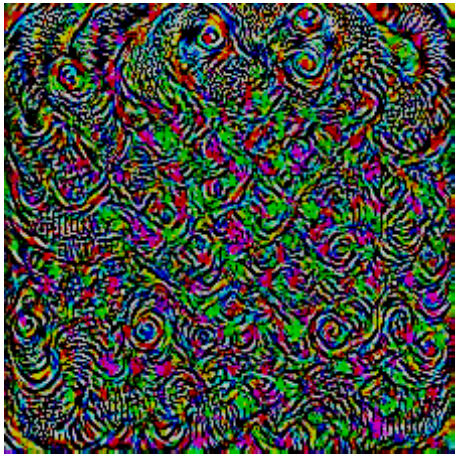
(b) VGG-F



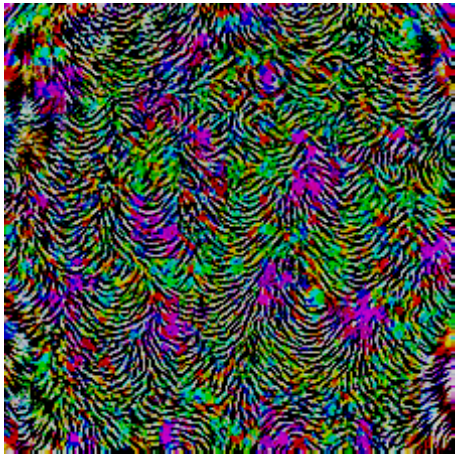
(c) VGG-16



(d) VGG-19



(e) GoogLeNet



(f) ResNet-152

Examples of Perturbed Images

Bouvier des Flandres

wool

Scottish deerhound

Indian elephant

Christmas stocking

Indian elephant

ski mask

African grey

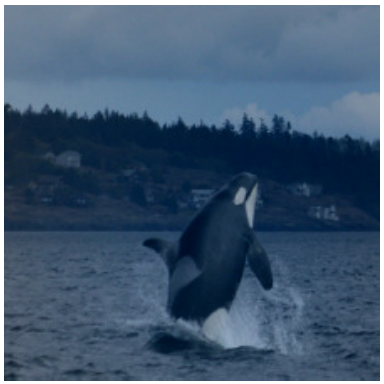
Examples of Perturbed Images

porcupine

tabby

European fire salamander

common newt



killer whale

African grey

toyshop

carousel

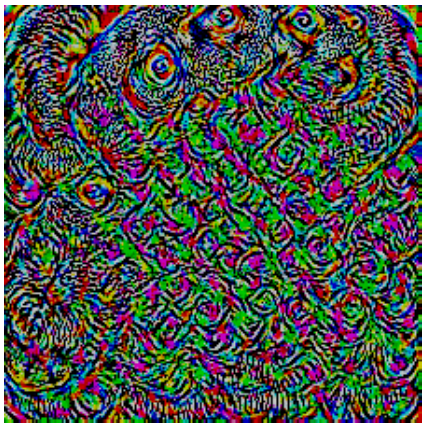
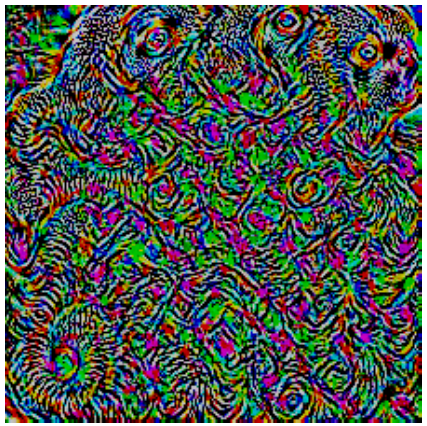
Cross-Model Universality (Doubly-Universal)

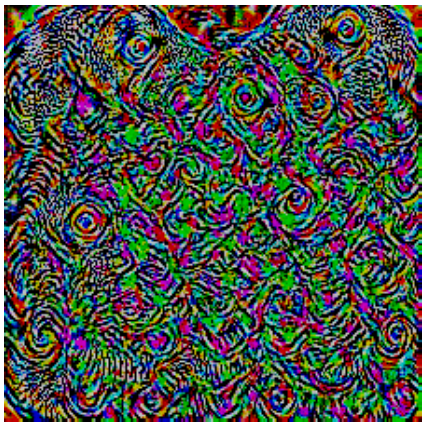
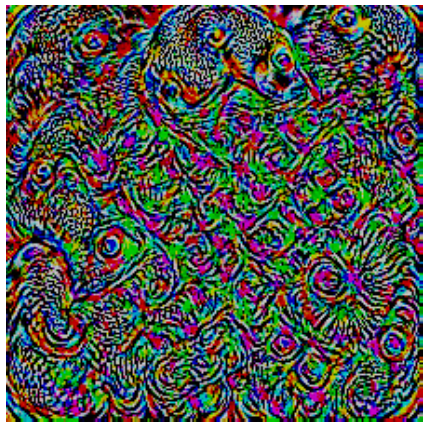
	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

Table: Generalizability of the universal perturbations across different networks. The percentages indicate the fooling rates. The rows indicate the architecture for which the universal perturbations is computed, and the columns indicate the architecture for which the fooling rate is reported.

Diversity of Universal Perturbations

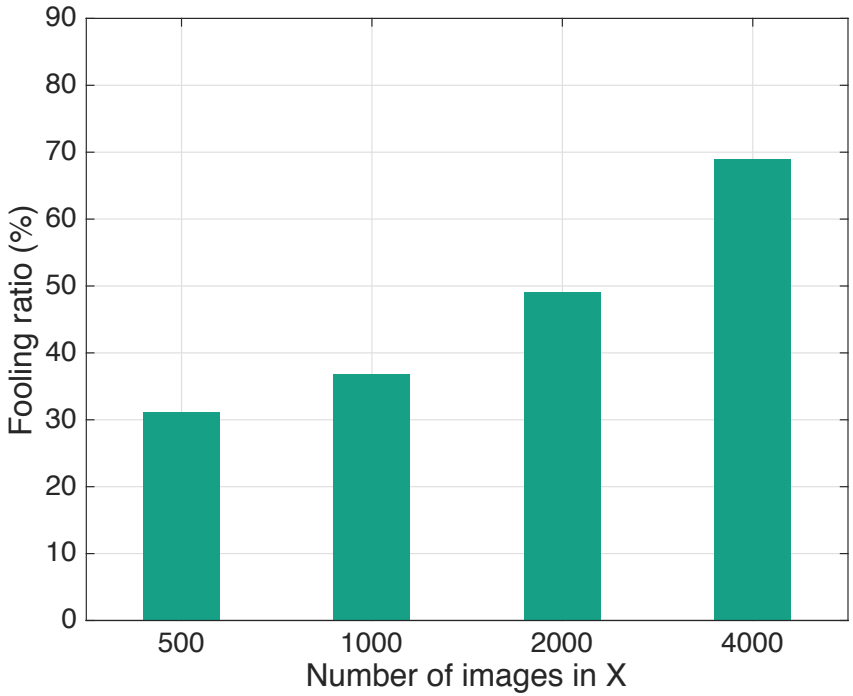
- The five perturbations are generated using different random shufflings of the set X.
- The normalized inner products for any pair of universal perturbations does not exceed 0.1!
- This highlights the diversity of such perturbations.

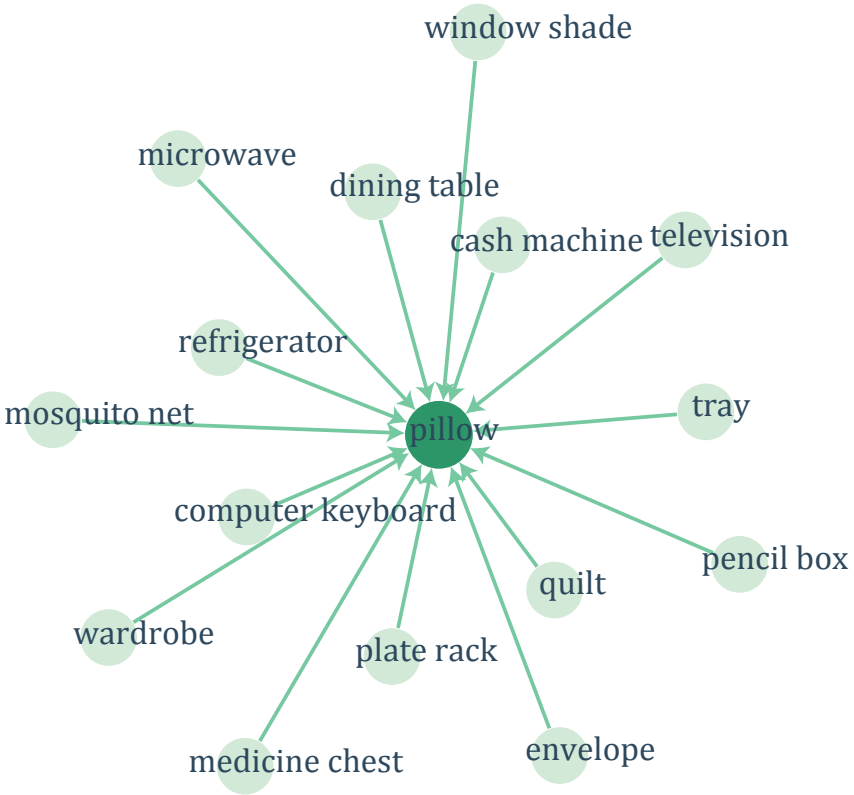



How many samples does it take to make a UAP?

- With a set X containing only 500 images, we can fool more than 30% of the images on the validation set.
- This result is significant when compared to the number of classes in ImageNet (1000), as it shows that we can **fool a large set of unseen images**, even when using a set X containing **less than one image per class!**



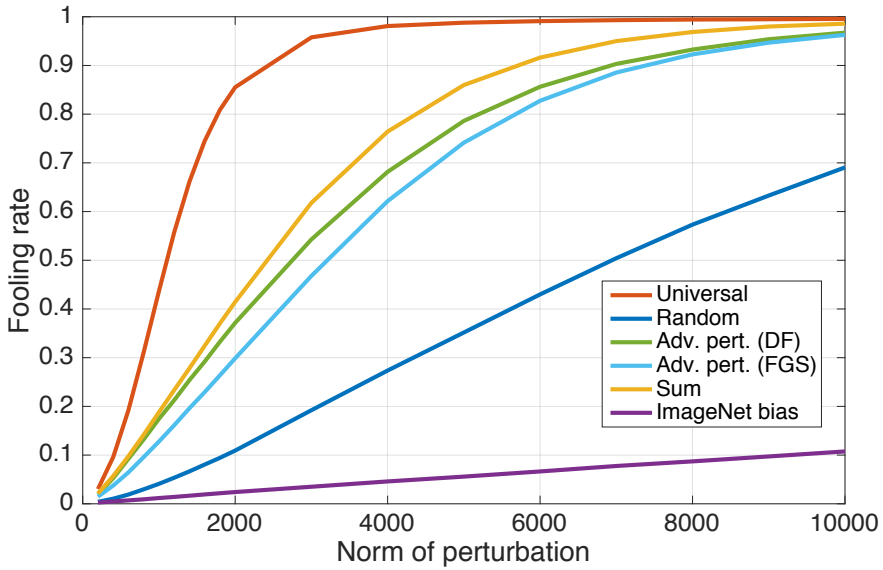
Relation Between Original and Perturbed Labels



Explaining the Vulnerability to Universal Perturbations

We compare universal perturbations with other types of perturbations:

- Random perturbation
- Adversarial perturbation computed for a randomly picked sample (computed using the DeepFool and FGSM)
- Sum of adversarial perturbations over X
- Mean of the images (or ImageNet bias).



The large difference between universal and random perturbations suggests that the universal perturbation exploits some **geometric correlations between different parts of the decision boundary** of the classifier.

- If the orientations of the decision boundary in the neighborhood of different data points were completely uncorrelated, the norm of the best universal perturbation would be comparable to that of a random perturbation.

Explaining the Vulnerability to Universal Perturbations

For each image x in the validation set, we compute the adversarial perturbation vector:

$$r(x) = \arg\min_r \|r\|_2 \quad \text{s.t.} \quad \hat{k}(x+r) \neq \hat{k}(x).$$

To quantify the correlation between different regions of the decision boundary of the classifier, we define the matrix:

$$N = \left[\frac{r(x_1)}{\|r(x_1)\|_2} \quad \cdots \quad \frac{r(x_n)}{\|r(x_n)\|_2} \right]$$

of normal vectors to the decision boundary in the vicinity of n data points in the validation set.

- For binary linear classifiers, the decision boundary is a hyperplane, and N is of rank 1, as all normal vectors are collinear.
- To capture more generally the correlations in the decision boundary of complex classifiers, we compute the singular values of the matrix N.
- We further show in the same figure the singular values obtained when the columns of N are sampled uniformly at random from the unit sphere.

Recall

Recall: Rank of a matrix

The rank of matrix A is the dimension of the vector space generated (or spanned) by its columns.

- This corresponds to the **maximal number of linearly independent columns** of A.
- The rank of A equals the number of non-zero singular values, which is the same as the number of non-zero diagonal elements in Σ in the singular value decomposition $A = U\Sigma V^\top$

Recall

Recall: Singular Value Decomposition (SVD)

The singular value decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. Consider a matrix A_{mn}, it can be uniquely decomposed as

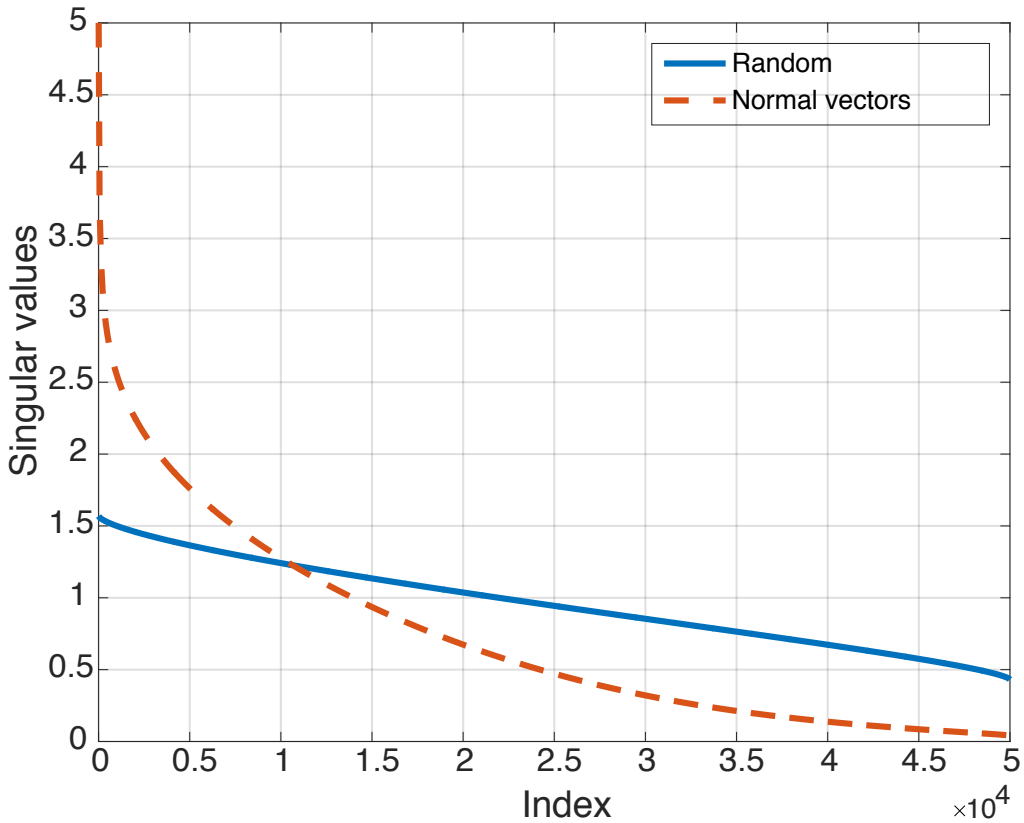
$$A_{mn} = U_{mm}\Sigma_{mn}V_{nn}^{\top} = \sum_{i=1}^{r} u_i \sigma_i v_i^{\top}$$

where U is an $m \times m$ matrix whose rows and columns are orthornormal (so $U^{\top}U = UU^{\top} = I_m$), V is $n \times n$ matrix whose rows and columns are orthonormal (so $V^{\top}V = VV^{\top} = I_n$), and Σ is a $m \times n$ matrix (a diagonal matrix) containing the $r = \min(m, n)$ singular values $\sigma_i \geq 0$ on the main diagonal, with 0s filling the rest of the matrix.
The singular values are the diagonal entries of the Σ matrix and are arranged in descending order. The columns of U are the left singular vectors, and the columns of V are the right singular vectors. Since the rank of matrix $u_i \sigma_i v_i^{\top}$ is 1 (all columns are a factor of u_i) and u_is are linearly independent, the rank of A equals the number of non-zero singular values.

Explaining the Vulnerability to UAPs

Observe that, while the latter singular values have a slow decay, the singular values of N decay quickly, which confirms the existence of large correlations and redundancies in the decision boundary of deep networks.

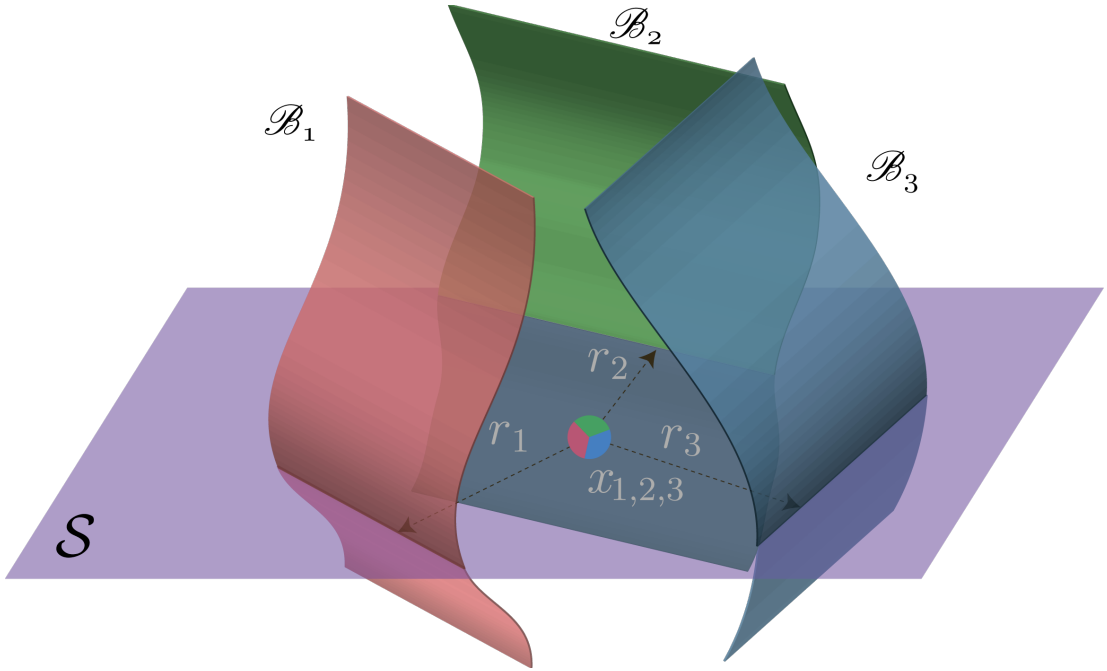
- More precisely, this suggests the existence of a subspace $\mathcal{S}$ of low dimension d' (with $d' \ll d$), that contains most normal vectors to the decision boundary in regions surrounding natural images.



Explaining the Vulnerability to UAPs

We hypothesize that the existence of universal perturbations fooling most natural images is partly due to the existence of such a low-dimensional subspace that captures the correlations among different regions of the decision boundary.

- To verify this hypothesis, we choose a random vector of norm $\xi = 2000$ belonging to the subspace $\mathcal{S}$ spanned by the first **100 singular vectors**.
- Such a perturbation can fool nearly 38% of these images, thereby showing that a random direction in this well-sought subspace $\mathcal{S}$ significantly outperforms random perturbations (random perturbations can only fool 10% of the data).



Unlike the above experiment, the proposed algorithm does not choose a random vector in this subspace, but rather chooses a specific direction in order to maximize the overall fooling rate. This explains the gap between the fooling rates obtained with the random vector strategy in $\mathcal{S}$ and Algorithm 2.

References and Resources I

- Madry et al. [3] introduced PGD as a multi-step first order adversary. They used this attack for their **Adversarial Training** which will be the topic of our next lecture.
- As of now, the **Auto-PGD (APGD)** attack by Croce and Hein [2] is considered state of the art.
- Moosavi-Dezfooli et al. [4] discovered the existence of Universal Adversarial Perturbations.
- Zou et al. [5] explore universal attacks on large language models (LLMs) which illustrates the significance and importance of these attacks.
- Brown et al. [1] were the first to introduce the concept of an adversarial patch.

References and Resources II

[1] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. **Adversarial Patch**, 2018. URL https://arxiv.org/abs/1712.09665.

[2] Francesco Croce and Matthias Hein. Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks, 2020. URL https://arxiv.org/abs/2003.01690.

[3] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. **Towards Deep Learning Models Resistant to Adversarial Attacks**, 2019. URL https://arxiv.org/abs/1706.06083.

[4] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. **Universal Adversarial Perturbations**, 2017. URL https://arxiv.org/abs/1610.08401.

[5] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models, 2023. URL https://arxiv.org/abs/2307.15043.