Robust Universal Adversarial Perturbations

Changming Xu
Department of Computer Science
University of Illinois Urbana-Champaign
Champaign, IL 61801
cxu23@illinois.edu

Gagandeep Singh
Department of Computer Science
University of Illinois Urbana-Champaign
Champaign, IL 61801
ggnds@illinois.edu

Abstract

Universal Adversarial Perturbations (UAPs) are imperceptible, image-agnostic vectors that cause deep neural networks (DNNs) to misclassify inputs with high probability. In practical attack scenarios, adversarial perturbations may undergo transformations such as changes in pixel intensity, scaling, etc. before being added to DNN inputs. Existing methods do not create UAPs robust to these real-world transformations, thereby limiting their applicability in practical attack scenarios. In this work, we introduce and formulate UAPs robust against real-world transformations. We build an iterative algorithm using probabilistic robustness bounds and construct such UAPs robust to transformations generated by composing arbitrary sub-differentiable transformation functions. We perform an extensive evaluation on the popular CIFAR-10 and ILSVRC 2012 datasets measuring our UAPs' robustness under a wide range common, real-world transformations such as rotation, contrast changes, etc. We further show that by using a set of primitive transformations our method can generalize well to unseen transformations such as fog, JPEG compression, etc. Our results show that our method can generate UAPs up to 23% more robust than state-of-the-art baselines.

1 Introduction

Deep neural networks (DNNs) have achieved impressive results in many application domains such as natural language processing [1, 11], medicine [18, 19], and computer vision [40, 42]. Despite their performance, they can be fragile in the face of adversarial perturbations: small imperceptible changes added to a correctly classified input that make a DNN misclassify. While there is a large amount of work on generating adversarial perturbations [41, 21, 35, 33, 12, 46, 16, 14, 44, 50, 5, 43], these works depend upon unrealistic assumptions about the power of the attacker: the attacker knows the DNN input in advance, generates input-specific perturbations in real-time and *exactly* combines the perturbation with the input before being processed by the DNN. Thus, we argue that these threat models are not realizable in many real-world applications.

Practically feasible adversarial perturbations. In this work, we consider a more practical adversary to reveal real-world vulnerabilities of state-of-the-art DNNs. We assume that the attacker (i) does not know the DNN inputs in advance, (ii) can only transmit additive adversarial perturbations, and (iii) their transmitted perturbations are susceptible to modification due to real-world effects. Examples of attacks in our threat model include adding stickers to the cameras for fooling image classifiers [31] or transmitting perturbations over the air for deceiving audio classifiers [30]. Note that our threat model is distinct from directly generating adversarial examples (i.e. creating physical adversarial objects [6]) which require access to the original input.

The first two requirements in our threat model can be fulfilled by generating Universal Adversarial Perturbations (UAPs) [36]. Here the attacker can train a single adversarial perturbation that has a high probability of being adversarial on all inputs in the training distribution. However, as our experimental

Preprint. Under review.

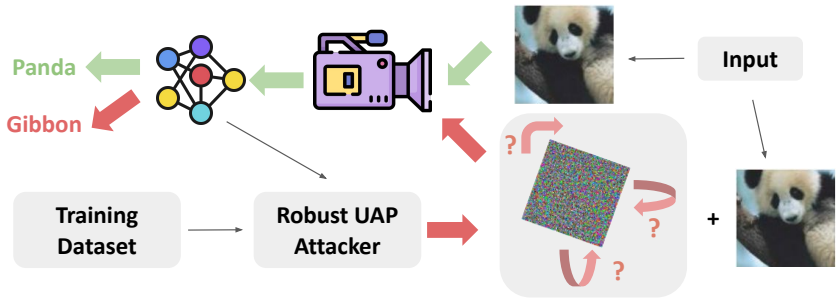

Figure 1: Robust UAP Threat Model: Input Agnostic + Robust to Transmission Transformation

results show, the generated UAPs need to be combined with the DNN inputs precisely, otherwise they fail to remain adversarial. In practice, changes to UAPs are likely due to real-world effects. For example, the stickers applied to a camera can undergo changes in contrast due to weather conditions or the transmitted perturbation in audio can change due to noise in the transmission channel. This non-robustness reduces the efficiency of practical attacks created with existing methods [36, 38, 31, 30].

This work: Robust UAPs. To overcome the above limitation, we propose the concept of robust UAPs: perturbations that have a high probability of remaining adversarial on inputs in the training distribution even after applying a set of real-world transformations. The optimization problem in generating robust UAPs [36] is made challenging as we are looking for perturbations that are adversarial for a set of inputs as well as to a set of potentially unknown transformations applied to the perturbations. To address this challenge, we make the following main **contributions**:

- We introduce *Robust UAPs* and formulate their generation as an optimization problem. We separate our threat model into two scenarios depending on whether the transformation set is known apriori.
- We design a new method, `RobustUAP`, for constructing robust UAPs. Our method is general and constructs UAPs robust to any transformations generated by composing arbitrary sub-differentiable transformation functions. We provide an algorithm for computing provable probabilistic bounds on the robustness of our UAPs against many practical transformations. We show that in the vision domain we can use a set of primitive transforms (adapted from Modas et al. [34]) to create *Universally* Robust UAPs.
- We perform an extensive evaluation of our method on state-of-the-art models for the popular CIFAR-10 [29] and ILSVRC 2012 [15] datasets. We compare the robustness of our UAPs under compositions of challenging real-world transformations, such as rotation, contrast change, etc. We show that on both datasets, the UAPs generated by `RobustUAP` are significantly more robust, achieving up to 23% more robustness, than the UAPs generated from the baselines.

Our work is complementary to the development of real-world attacks [30, 31] in various domains, which require modeling how the universal perturbations change during transmission. `RobustUAP` can improve the efficiency of such attacks by constructing perturbations that are more robust to real-world transformations than with existing algorithms [36, 38, 30, 31]. Our results using primitive transformations in vision suggest that we can forego domain specific modeling in other domains if we can find a good set of primitives for that domain.

2 Background

In this section, we provide necessary background definitions and notation used in the rest of our work. For the remainder of the paper, let $\mu \subset \mathbb{R}^d$ be the input data distribution, $\mathbf{x} \in \mu$ be an input point with the corresponding true label $y \in \mathbb{R}$, and $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ be our target classifier. For ease of notation, we define $f_k(\mathbf{x})$ to be the k^{th} element of $f(\mathbf{x})$ and allow $\hat{f}(\mathbf{x}) = \arg\max_k f_k(\mathbf{x})$ to directly refer to the classification label. We use $\mathbf{v}$ to reference image specific perturbations and $\mathbf{u}$ to reference universal adversarial perturbations, $\mathbf{v_r}$ and $\mathbf{u_r}$ refer to the robust variants and will be defined in Sec. 3. We provide formal definitions in Appendix A.

Adversarial Examples and Perturbations. An *adversarial example* is a misclassified data point that is *close* (in some norm) to a correctly classified data point [21, 33, 12]. In this paper, we consider examples $\mathbf{x}'$ generated as $\mathbf{x}' = \mathbf{x} + \mathbf{v}$ where $\mathbf{v}$ is an *adversarial perturbation*.

Universal Adversarial Perturbations. UAPs are single vector, input-agnostic perturbations [36]. They differ from traditional adversarial attacks, which create perturbations dependent on each input sample. To measure UAP performance, we introduce the notion of universal adversarial success rate (ASR_U), which measures the probability that a perturbation $\mathbf{u}$ when added to $\mathbf{x}$, sampled from μ, causes a change in classification under f. Thus a perturbation, u, is a UAP given two conditions: its ASR_U is greater than a given threshold, γ, and its norm is small. If an additive perturbation has a small l_p-normit does not affect the semantic content of the image as it appears as noise. We pose the construction of UAPs as an expectation minimization problem:

$$\arg\min_{u} \mathbb{E}_{\mathbf{x}\sim\mu}[\delta(\hat{f}(\mathbf{x}+\mathbf{u}), \hat{f}(\mathbf{x}))] \text{ s.t. } ||\mathbf{u}||_p < \epsilon \tag{1}$$

where δ is the Kronecker Delta function [2].

Figure 2: Robust UAPs (left) cause a classier to misclassify on *most* of the data distribution even after transformations are applied on them. Standard UAPs (right) are not robust to transformations and have a low probability of remaining UAPs after transformation.

3 Robust Universal Adversarial Perturbations

In this section, we first define our notion of transformation sets and neighborhoods in order to define robust UAPs. Here, when we are referencing transformation sets as the ones applied during transmission, if these are unknown, we detail our method for overcoming this in Section 4.1. Formal definitions of all terms can be found in Appendix A.

Transformation Sets and Neighborhoods. We define a transformation set, T, as all transforms, τ, which can be made by composing from a predefined set of bijective sub-differentiable transformation functions. The neighborhood, $N_T(v)$, of a point v is all points, v' reachable from v using transformations from T.

Example 3.1. *Let T be all transformations represented by a rotation of $\pm 30°$ and scaling of up to a factor of 2, in this case one $\tau \in T$ could be {rotation of $8°$ and scaling a factor of 1.2} in that order and $N_T(\mathbf{v})$ would include any point obtained by applying a transformation from T on $\mathbf{v}$.*

Robust UAPs. In order to define robust UAPs we introduce robust universal adversarial success rate. The *robust universal adversarial success rate*, ASR_R, measures the probability that a neighbor of $\mathbf{u_r}$ is also an UAP on μ, i.e. after transformation it maintains high universal ASR above some threshold γ. We note that even though $||\mathbf{u_r}||_p \leq \epsilon$, it can happen that a $\mathbf{u'_r} \in N_T(\mathbf{u_r})$ has $||\mathbf{u'_r}||_p > \epsilon$. Therefore, we require that the norm of $\mathbf{u'_r}$ is small.

Definition 3.2. *A robust UAP, $\mathbf{u_r}$, is one which most points within a neighborhood of $\mathbf{u_r}$ when added to most points in μ fool the classifier, f. $\mathbf{u_r}$ satisfies $||\mathbf{u_r}||_p < \epsilon$ and $ASR_R(f, \mu, T, \gamma, \mathbf{u_r}) > \zeta$.*

In order to construct robust UAPs, we can pose the following expectation minimization problem:

$$\arg\min_{\mathbf{u_r}} \mathop{\mathbb{E}}_{\mathbf{u'_r}\in N_T(\mathbf{u_r})} [I(||\mathbf{u'_r}|| < \epsilon) \times \mathop{\mathbb{E}}_{\mathbf{x}\sim\mu} [\delta(\hat{f}(\mathbf{x}+\mathbf{u'_r}), \hat{f}(\mathbf{x}))]] \text{ s.t. } ||\mathbf{u_r}||_p < \epsilon \tag{2}$$

Here $I : \mathbb{R}^d \rightarrow \mathbb{R}$ denotes an indicator function. The inner expectation represents the UAP condition for the transformed perturbation $\mathbf{u_r'}$ while the outer expectation represents the neighborhood robustness condition. The composition of these conditions in Equation 2 makes it computationally harder than minimizing over only the transformation set, as in EOT [6], or than minimizing over only the data distribution, as in Equation 1.

4 Generating Robust Universal Adversarial Perturbations

In this section we discuss how we deal with both known and unknown transformation sets, then describe our approach for optimizing Equation 2. As it would be computationally prohibitive to precisely compute the expected value, we estimate the expected value per batch, $\hat{\mathbf{x}} \subset \mu$, and random set of transformations sampled from T, $\hat{\tau} \subset T$. We can approximate Equation 2 using:

$$\frac{I(||\hat{\tau}_j(\mathbf{u_r})|| < \epsilon)}{|\hat{\mathbf{x}}| \times |\hat{\tau}|} \sum_{i=1}^{|\hat{\mathbf{x}}|} \sum_{j=1}^{|\hat{\tau}|} L[f(\hat{\mathbf{x}}_\mathbf{i} + \hat{\tau}_j(\mathbf{u_r})), f(\hat{\mathbf{x}}_\mathbf{i})] - \lambda||\mathbf{u_r}||_p \tag{3}$$

We describe our two threat models and some intuitive baselines for optimizing Equation 2. We then present our new algorithm, `RobustUAP`.

4.1 Threat Models: Known vs Unknown Transformation Sets

In the above section, we have assumed that the transformations applied during transmission is known to the attacker and used to train the UAP. However, in a real-world attack scenario the attacker may not know precisely what transformations its perturbation will undergo. In such scenarios, they may want their attack to be robust to unseen perturbations. In this case, we propose generating robust UAPs using a set of primitive transformations. For the image domain, we draw from existing work in the data augmentation area. In their paper, **PRI**mitives of **M**aximum **E**ntropy (PRIME), Modas et al. [34] define three primitive transformations: spectral, spatial, and color. Using random combinations of these transformations to train, they find that they are able to generalize well to unseen transformations such as frost, JPEG compression, motion blur, etc. We can use these primitive transformations to generate robust UAPs and we show that in practice this generates robust UAPs which generalize well to a variety of unseen transforms. In other domains, we hope that this work helps to inspire finding similar primitive transformation sets.

4.2 Baseline Algorithms

We propose two baseline algorithms for generating robust UAPs. The first method is momemtum based Stochastic Gradient Descent (SGD). We can directly solve Equation 2 using gradient descent. The second baseline is leveraging the standard UAP algorithm from Moosavi-Dezfooli et al. [36], but instead of computing an adversarial perturbation at each point, we compute a robust adversarial perturbation at each point. More details about both of these baseline algorithms can be found in Appendix E. Both of these algorithms can be seen as naively combining the EoT and UAP algorithms, in the next section we describe `RobustUAP` our algorithm which takes a more principled approach at robust UAP generation.

4.3 Robust UAP Algorithm

The baseline algorithms have two fundamental limitations: (i) they rely on random sampling over the symbolic transformation region, but the sampling strategy does not explicitly try to maximize the robustness of the generated UAP over the entire symbolic region, and (ii) they do not estimate robustness on unsampled transformations. These baselines can be seen as naive combinations of the EoT and UAP algorithms. As a result, the baselines yield suboptimal UAPs (as confirmed by our experiments below). To overcome these fundamental limitations, we create a method to compute probabilistic bounds for expected robustness on an entire symbolic region. We leverage this method for approximating expected robustness in a new algorithm to generate robust UAPs with guarantees. We make a simplifying assumption that $N_T(\mathbf{u_r})$ has a well defined, sampleable probability density function (PDF) as we cannot bound robustness for arbitrary transformations. Our experiments show that even though our assumptions do not hold for all the transformation sets considered in this work, they significantly improve the robustness of our generated UAPs. Our approximation of the expected robustness relies on the following theoretical result:

Theorem 4.1. *Given a perturbation $\mathbf{u_r}$, a neural network f, a finite set of inputs $\mathbf{X}$, a set of transformations T, and minimum universal adversarial success rate $\gamma \in \mathbb{R}$. Let $p(\gamma) =$*

$P_{\mathbf{u}'_\mathbf{r} \sim N_T(\mathbf{u_r})}(ASR_U(f, \mathbf{X}, \mathbf{u}'_\mathbf{r}) > \gamma)$. *For* $i \in 1 \dots n$, *let* $\mathbf{u}^\mathbf{i}_\mathbf{r} \sim N_T(\mathbf{u_r})$ *be random variables with a well defined PDF and* $I : \mathbb{R}^d \to \mathbb{R}$ *be the indicator function, let*

$$\hat{p}_n(\gamma) = \frac{1}{n} \sum_{i=1}^{n} I(ASR_U(f, \mathbf{X}, \mathbf{u}^\mathbf{i}_\mathbf{r}) > \gamma) \tag{4}$$

For accuracy level, $\psi \in (0, 1)$*, and confidence,* $\phi \in (0, 1)$*, where* $(0, 1)$ *is the open interval between* 0 *and* 1*. If* $n \geq \frac{1}{2\psi^2} \ln \frac{2}{\phi}$ *then*

$$P(|\hat{p}_n(\gamma) - p(\gamma)| < \psi) \geq 1 - \phi \tag{5}$$

The proof of this theorem can be found in Appendix B. Theorem 4.1 states that with enough samples from the neighborhood of a perturbation, $\mathbf{u_r}$, the adversarial success rate of $\mathbf{u_r}$ on the entire neighborhood is arbitrarily close to the adversarial success rate of $\mathbf{u_r}$ on sampled transformations with probability greater than $1 - \phi$.

Leveraging Theorem 4.1, we create `EstRobustness` which given accuracy, ψ, and confidence, ϕ, returns the ASR_R on a finite set of inputs with probabilistic robustness guarantees under the assumptions of Theorem 4.1. The pseudocode for `EstRobustness` is in Appendix H.

Our algorithm: `RobustUAP`**.** We leverage Theorem 4.1 and `EstimateRobustness` to develop `RobustUAP`, the pseudocode for which is seen in Algorithm 1. Similar to the SGD baseline, we approximate the expectation in Equation 2 in batches. We first sample transformations from the PDF of the neighborhood. We set the number of transformations, n, based on Theorem 4.1 to satisfy the desired confidence level and accuracy. For each gradient step, we compute the mean loss over the current batch and set of sampled transforms (line 8). For each set of batch and sampled transformations, instead of making a single gradient update like SGD, we use Projected Gradient Descent (PGD) to iteratively compute a more robust update to the universal perturbation and end only when the estimated robustness on the batch satisfies a given threshold (line 10). At the end of each epoch, we check the robustness across the entire training set and transformation space using `EstRobustness` (`ER`) and stop when we have reached the desired performance (line 14).

Algorithm 1 Robust UAP Algorithm

1: Initialize $\mathbf{u_r} \leftarrow 0, n \leftarrow \lceil \frac{1}{2\psi^2} \ln \frac{2}{\phi} \rceil$
2: **repeat**
3: **for** $\mathbf{B} \subset \mathbf{X}$ **do**
4: For $i = 1 \dots n$ sample $\tau_i \sim T$
5: **if** $\texttt{ER}(f, \mathbf{B}, T, \gamma, \mathbf{u_r}, \psi, \phi) < \zeta$ **then**
6: $\Delta\mathbf{u_r} \leftarrow 0$
7: **repeat**
8: Compute $L_{\mathbf{B},\tau} = \frac{1}{|\mathbf{B}| \times n} \sum_{i=1}^{|\mathbf{B}|} \sum_{j=1}^{n} L[f(\mathbf{B_i} + \tau_j(\mathbf{u_r} + \Delta\mathbf{u_r})), f(\mathbf{B_i})]$
9: $\Delta\mathbf{u_r} = \mathcal{P}_{p,\epsilon}(\Delta\mathbf{u_r} + \alpha \text{sign}(\nabla L_{\mathbf{B},\tau}))$
10: **until** $\texttt{ER}(f, \mathbf{B}, T, \gamma, \mathbf{u_r} + \Delta\mathbf{u_r}, \psi, \phi) < \zeta$
11: $\mathbf{u_r} \leftarrow \mathcal{P}_{p,\epsilon}(\mathbf{u_r} + \Delta\mathbf{u_r})$
12: **end if**
13: **end for**
14: **until** $\texttt{ER}(f, \mathbf{X}, T, \gamma, \mathbf{u_r}, \psi, \phi) < \zeta$

5 Evaluation

Our `RobustUAP` framework is applicable to all transformation sets in a variety of domains. We empirically evaluate our method `RobustUAP` and three baseline approaches (`SGD`, `StandardUAP_RP`, `StandardUAP` [36]) on popular models from the vision domain. We show that `RobustUAP` is more robust on both uniform random noise and compositions of real-world transformations such as rotation, scaling, etc. We did not have the hardware to print high resolution transparent stickers so we could not produce real-world results in the vision domain. We show that training `RobustUAP` on a set of primitive transforms results in a universally robust UAP which generalizes well to unseen transformations allowing for successful attacks without the need for domain specific modeling.

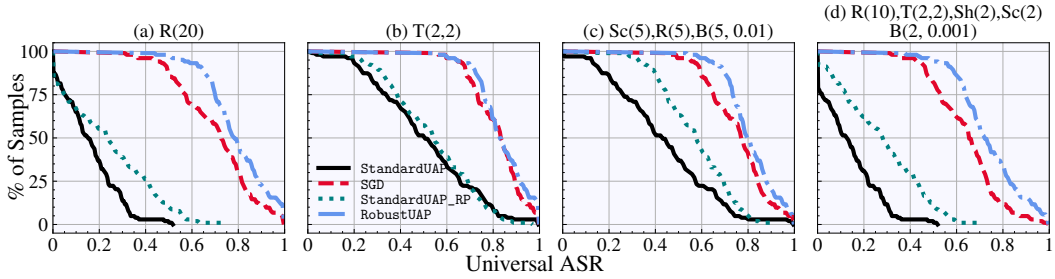

Figure 3: For each method, a point (x, y) in the corresponding line represents the percentage of sampled UAPs ($y\%$) with Universal ASR $> x$ for the different semantic transformations on ILSVRC.

Experimental evaluation. We consider two popular image recognition datasets: CIFAR-10[29] and ILSVRC 2012[15]. We evaluate on a pretrained VGG16 [40] and Inception-v3 [42] network on CIFAR-10 and ILSVRC 2012 respectively. For both we evaluate on a random subset (1000 images) for the test set. All experiments were performed on a desktop PC with a GeForce RTX(TM) 3090 GPU and a 16-core Intel(R) Core(TM) i9-9900KS CPU @ 4.00GHz.

We report the results for l_2-norm with $\epsilon = 100$ for ILSVRC 2012 and $\epsilon = 10$ for CIFAR-10. These values were chosen based on the values presented by the original UAP paper [36]. We use an image normalization function given by our pretrained models and thus scaled our ϵ values accordingly. We note that the ϵ-values are significantly smaller than the image norms resulting in imperceptible perturbations that do not affect the semantic content of the image. Due to the hardness of the optimization problem, for the same norm value, the effectiveness of a UAP is less than input-specific perturbations; however, crafting input-specific perturbations requires making unrealistic assumptions about the power of the attacker as mentioned in the introduction and therefore we do not consider them part of our threat model which aims to generate practically feasible perturbations. We use $\psi = 0.05$ and $\phi = 0.05$ resulting in $n = 738$ for generating samples for our `RobustUAP` algorithm as well as reporting robust ASR in our evaluation. The UAPs are trained on 2,000 images, other parameters for evaluation are given in Appendix I. Error bars/variances are reported in Appendix Z.

5.1 Robustness to Random Noise

We first generate UAPs robust against uniform random noise. Here since our neighborhood has a well-defined PDF we get the robustness guarantees from `EstimateRobustness`, in the following sections when we consider semantic and unknown transformations we won't have the same gaurantees. The results and future discussion can be found in Appendix M.

5.2 Robustness to Semantic Transformations

Next, we consider transformation sets generated by composing five popular semantic transformations in existing literature [6, 8]: brightness/contrast, rotation, scaling, shearing, and translation.

We use a variety of different compositions to show that our algorithm works under different conditions, and base our parameters for the transformations on [8]. For our experiments, $R(\theta)$ corresponds to rotations with angles between $\pm\theta$; $T(x, y)$, to translations of $\pm x$ horizontally and $\pm y$ vertically; $Sc(p)$ to scaling the image between $\pm p\%$; $Sh(m)$ to shearing by shearing factor between $\pm m\%$; and $B(\alpha, \beta)$ to changes in contrast between $\pm\alpha\%$ and brightness between $\pm\beta$. Further details about these transformations can be seen in Appendix C. We consider compositions of different subsets and ranges of these transformations shown in Table 1 including composing all transformations together. The hardness of generating robust UAPs depends on the effect that the transformation set has on the UAP (i.e. random noise has a relatively small effect compared to rotation). The hardness also increases with the number of transformations in the composition as well as the range of parameters for each individual transformation. For example, generating robust UAPs is harder for the composition shown in the first and last row for ILSVRC 2012 in Table 1 compared to the second and third row. The same is true for generating a UAP robust to uniform random noise.

Robust ASR ($\mathbf{ASR_R}$). Figure 3 shows performance of UAPs obtained by applying 738 randomly sampled transformations to the original UAPs generated by different methods on ILSVRC, similar graphs for CIFAR-10 can be found in Appendix K. The `RobustUAP` algorithm outperforms all others in each case, we observe that for these harder transformation sets `StandardUAP` loses its effectiveness completely. In Table 1 we compare robust universal adversarial success rate ASR_R with $\gamma = 0.6$, in other words, we are finding the percentage of sampled neighbors of the perturbation that are still UAPs with 60% effectiveness on the testing set. We provide average ASR_U scores as well as ASR_R for different γ levels in Appendix L.

Our RobustUAP algorithm achieves at least 53.4% higher robust ASR when compared to the standard UAP algorithm on both datasets and all transformation sets. Furthermore, our RobustUAP algorithm significantly outperforms both robust baseline approaches. Except for the $T(2,2)$ case which we observe to be the easiest, RobustUAP achieves at least 11.6% performance gain over the baselines. SGD is the best performing baseline and achieves high robust ASR on relatively easier transformation sets performing within 1% of RobustUAP on $T(2,2)$. On harder transformation sets, this gap widens considerably, see Table 1.

5.3 Universally Robust UAPs

Using the set of primitive transformations discussed by PRIME, we generate robust UAPs on ILSVRC using the same parameters as above. For each sampled transform, we randomly apply three transformations from identity, spectral, spatial, and color. This means that we can get multiple of the same transformation or even no transformation. We follow the setup from PRIME for the parameters of each transformation type. Table 2 shows the robust ASR when training a RobustUAP on PRIME, Affine ($R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$), and Fog transformation sets and how they perform on common corruptions [34, 27, 24]. Although prime does not inherently contain any specific affine or common corruption in its training it has generally high robustness (>58.3%) against all transformation sets tested. We observe that training on the target transformation set does bring higher robustness than training on PRIME (i.e. Affine-trained robust UAP has best performance on Gaussian, Contrast, Affine while Fog-trained robust UAP has best performance on fog); however, we find that PRIME has much better performance on unseen transformations (i.e. Fog-trained or Affine-trained robust UAP on JPEG). Our results suggest that a set of good primitive transformations is sufficient for generating universally robust UAPs that generalize well to unseen transformations.

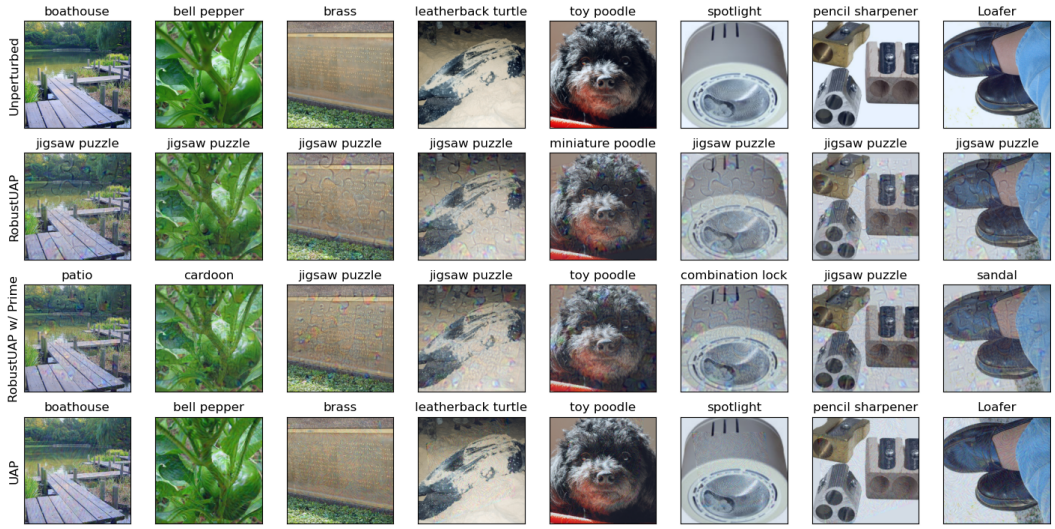

Figure 4: Examples of perturbed images with labels. The top row is unperturbed ILSVRC 2012 test set images, the second row has a randomly transformed robust UAP added to it, the third row has a randomly transformed robust UAP trained with Prime added to it, and the bottom row has a randomly transformed standard UAP added to it. Labels calculated using Inception-v3.

Table 1: Robust ASR of RobustUAP compared to the three baselines.

Dataset	Transformation Set	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC 2012	$R(20)$	0.0%	69.9%	2.9%	$\mathbf{93.2}\%$
	$T(2,2)$	35.9%	96.1%	38.8%	$\mathbf{97.1}\%$
	$Sc(5), R(5), B(5,0.01)$	22.3%	85.4%	43.7%	$\mathbf{96.1}\%$
	$R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$	0.0%	63.1%	2.9%	$\mathbf{86.4}\%$
CIFAR-10	$R(30), B(2,0.001)$	0.0%	64.1%	2.9%	$\mathbf{75.7}\%$
	$R(2), Sh(2)$	42.7%	88.3%	52.4%	$\mathbf{96.1}\%$
	$R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$	0.0%	58.3%	7.8%	$\mathbf{79.6}\%$

Table 2: Robust ASR (%) of `RobustUAP` trained on PRIME, Affine ($R(10)$, $T(2,2)$, $Sh(2)$, $Sc(2)$, $B(2,0.001)$), and Fog when applied to Prime, Affine, and common corruption transforms

	Evaluation Corruption Set															
	Noise					Blur				Weather				Digital		
Train Set	Prime	Aff.	Gaus.	Shot	Imp.	Defo.	Glass	Moti.	Zoom	Snow	Fog	Frost	Bright	Contr.	Elast.	Pixel	JPEG
PRIME	**68.4**	58.3	72.1	81.3	**88.6**	**66.5**	**75.2**	**81.0**	74.6	**77.8**	78.8	**65.3**	**85.3**	90.4	**74.2**	**69.2**	**73.3**
Affine	10.1	**86.4**	**91.2**	**93.2**	85.4	45.1	31.4	76.1	**92.4**	65.2	70.1	50.1	80.1	**94.1**	39.1	30.5	37.3
Fog	1.5	0.1	10.2	11.3	9.3	15.1	7.6	10.1	5.5	69.5	**95.2**	21.3	10.1	12.6	18.4	2.8	3.9

5.4 Visualization

We visualize UAPs generated with different algorithms transformed randomly from $R(10)$, $T(2,2)$, $Sh(2)$, $Sc(2)$, $B(2,0.001)$ and added to images in ILSVRC 2012 in Figure 4. Our robust UAPs have a similar level of imperceptibility to standard UAPs. Robust UAPs affect the model classification after transformation with high probability, unlike standard UAPs. We further visualize UAPs generated with our three robust algorithms on the same transformation set and dataset in Appendix Y.

5.5 Transferability of Robust UAPs

We evaluate the transferability of `RobustUAP`. Previous works on UAPs [36] show that UAPs are transferable across different models. Here, we will evaluate whether robust UAPs exhibit the same behavior for robustness. The robust UAPs studied here are generated with `RobustUAP` on $R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$ for ILSVRC-2012 with $\gamma = 0.6$. We use a variety of models: Inception-v3 [42], MobileNet [25], Inception-v3 trained to be robust on $R(20)$ (InceptionR20), Inception-v3 trained to be robust on horizontal flips (InceptionHF), and ViT [17]. Table 3 shows us that our robust UAPs are transferable between different architectures. Our results show that robust UAPs transfer their robustness properties between architectures and models. Ignoring ViT, on all of the Inception and MobileNet models, the generated UAPs maintain at least 65% robust ASR when transferred to each other. This transfer is less but still significant for ViT where it maintains at least 32% robustness when transferred to or from the other models.

Table 3: Robust ASR when UAP is learned on source model and transfered to target model.

	Target Model				
Source Model	Inception	MobileNet	InceptionR20	InceptionHF	ViT
Inception	**86.4**%	65.2%	75.2%	78.5%	35.1%
MobileNet	74.3%	**86.2**%	67.3%	68.6%	38.3%
InceptionR20	80.1%	67.3%	**81.3**%	73.1%	32.0%
InceptionHF	77.8%	70.9%	75.8%	**83.8**%	34.6%
ViT	41.2%	32.4%	43.2%	39.7%	**88.5**%

5.6 Robustness against Robust UAPs

Traditional methods for robustness, such as adversarial training, focus on being robust in scenarios where the attacker is powerful (i.e. PGD), but with this comes a significant tradeoff in accuracy. In a preliminary study of practical robustness, we train for robustness against practical attacks such as Robust UAP, while maintaining high accuracy and faster training times. We perform our experiments on CIFAR-10, with a VGG16 model architecture, and with our most challenging transformation set ($R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$). We use a batch-wise variant of our robust UAP algorithm to do adversarial training. With this training method, our model obtains a Robust ASR of 0.4% and an accuracy of 90.1%. In contrast, standard adversarial training obtains a Robust ASR of 0.2% and an accuracy of 83.5%. Here, we can see that our training method obtains almost the same practical robustness without a significant reduction in accuracy. Further, our adversarial training method with robust UAP takes 12 minutes while standard adversarial training takes 48 minutes, which is one indication that our proposed training method is more efficient. We believe that further study can find improvement in robustness, accuracy, and efficiency of this type of training method which allows us to be practically robust while sacrificing less accuracy.

5.7 Additional Experiments

In Appendix N we show how our robust UAPs compare to standard UAPs on the non-robust universal ASR metric. In Appendix O, we evaluate our methods on ResNet18 [22] and MobileNet [25] for CIFAR-10 and ILSVRC 2012 respectively. The results follow the same trends as those reported in Table 1. To address additional real-world transformations, we investigate fog perturbations from [27]

in Appendix P finding similar results. We show that our methods work well in the targeted attack domain as well, results can be see in Appendix S. In Appendix U, we show that `RobustUAP` has good data efficiency and obtains good performance with less data. In Appendix T, we find that `RobustUAP` beats out `SGD` even at the same runtime. The recent popularity of transformer based models has also led us to show that our methods work on transformer based networks, results in Appendix V. In Appendix W, we find that traditional model robustness does not seem to effect ability to create robust UAPs. Finally, in Appendix X, we perform an ablation study on optimization strategy and show that SGD outperforms other popular optimizers.

6 Related Work

UAP Algorithms. Most works focusing on UAPs [36, 37, 48, 28, 3, 23, 49] generate singular vectors and do not consider perturbation robustness. Bahramali et al. [7] introduces a perturbation generator model (PGM) for the wireless domain which creates UAPs. They show that both adversarial training and noise subtracting defenses used in the wireless domain are highly effective in mitigating the effects of a single vector UAP attack; they further show that their method of generating a set of UAPs is an effective way for an attacker to circumvent these defenses. Although PGM provides a method for efficiently sampling unique UAPs, it is not robust to real-world transformations. In contrast, our method enables efficient sampling of UAPs that are robust to transformations.

Robust Adversarial Examples. The following papers introduce notions of robustness under different viewpoints and environmental conditions for constructing realizable adversarial examples. This is a different threat model compared to the additive perturbations discussed in this paper. Luo et al. [32] constructs adversarial examples which minimize human detectability, further introducing the idea of robustness for adversarial examples. They show that their attacks are robust against jpeg compression. Sharif et al. [39] attack facial recognition systems by putting adversarial perturbations on glass frames. Their work demonstrates a successful physical attack under stable conditions and poses. Eykholt et al. [20] proposes Robust Physical Perturbations (RP_2) in order to show that adding graffiti on a stop sign can cause it to be misclassified in both simulations and in the real world. Athalye et al. [6] introduce Expectation over Transformation (EOT) and use it to print real-world objects which are adversarial given a range of physical and environmental conditions.

Robust Adversarial Perturbations. Li et al. [30] generates music which affects a voice assistant based system from picking up its wake word. Li et al. [31] presents a method for generating a targeted adversarial sticker which changes an image classifier's classification from one pre-specified class to another. Both of these methods rely on specific use cases and are tailored towards generating adversaries coming from strict distributions, e.g. [30] generates guitar music while [31] generates a small grid of dots. These works build on algorithms akin to our baseline approaches and are limited in scope to domain specific transformations. Our work provides a framework for improving robustness against a wide range of transformations in diverse domains and can be leveraged for improving the effectiveness of these attacks.

7 Limitations and Ethics

We have shown the benefits of `RobustUAP` and would like to outline a few limitations and ethical concerns. Firstly, we note that our methods do not have a way to address non-differentiable transformations. We hope that future work leverges methods such as REINFORCE which do not have dependence on differentiability [45]. Secondly, `RobustUAP` would not work against models trained with standard adversarial training since to have a UAP you need to be able to generate standard adversarial examples. However, currently, robustly trained models are seldom used since adversarial attacks are hard to realize (i.e. UAPs) and these models come with a large tradeoff with accuracy. Our preliminary research suggests that defending against practical attacks such as robust UAP does not come with the same tradeoff, allowing for more practical robustness while retaining similar accuracy. We understand that our proposed methods could cause harm to existing deployed ML methods. Our hope is to expose vulnerabilities in existing safety and security critical applications of ML to spawn further research into practical robustness against real-world implementable attacks.

8 Conclusion

In this paper, we demonstrate that standard UAPs fail to be universally adversarial under transformation. We propose a new method, `RobustUAP`, to generate robust UAPs based upon obtaining probabilistic bounds on UAP robustness across an entire transformation space. We show that

`RobustUAP` works for both known and unknown transformation sets. Our experiments provide empirical evidence that our principled approach generates UAPs that are more robust than those from the existing/baseline methods. Our preliminary work suggests that robustness against practical adversaries such as robust UAPs may require much less tradeoff with accuracy and we hope that inspires research into robustness against practical attacks.

References

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] DC Agarwal. *Tensor Calculus and Riemannian Geometry*. Krishna Prakashan Media, 2013.

[3] Naveed Akhtar, Jian Liu, and Ajmal Mian. Defense against universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3389–3398, 2018.

[4] Cesare Alippi. *Intelligence for embedded systems*. Springer, 2014.

[5] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search, 2019.

[6] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[7] Alireza Bahramali, Milad Nasr, Amir Houmansadr, Dennis Goeckel, and Don Towsley. Robust adversarial attacks against dnn-based wireless communication systems. *arXiv preprint arXiv:2102.00918*, 2021.

[8] Mislav Balunović, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin Vechev. Certifying geometric robustness of neural networks. *Advances in Neural Information Processing Systems 32*, 2019.

[9] Philipp Benz, Chaoning Zhang, Tooba Imtiaz, and In So Kweon. Double targeted universal adversarial perturbations. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[10] Philipp Benz, Soomin Ham, Chaoning Zhang, Adil Karjauv, and In So Kweon. Adversarial robustness comparison of vision transformer and mlp-mixer to cnns. *arXiv preprint arXiv:2110.02797*, 2021.

[11] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[12] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[13] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[14] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack, 2019.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[18] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[19] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[20] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[21] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[22] Kaiming He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition." computer vision and pattern recognition (2015). *Google Scholar There is no corresponding record for this reference*, pages 770–778, 2015.

[23] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 2755–2764, 2017.

[24] Dan Hendrycks and Thomas G Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018.

[25] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[26] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015.

[27] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18963–18974, 2022.

[28] Valentin Khrulkov and Ivan Oseledets. Art of singular vectors and universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8562–8570, 2018.

[29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[30] Juncheng Li, Shuhui Qu, Xinjian Li, Joseph Szurley, J. Zico Kolter, and Florian Metze. Adversarial music: Real world audio adversary against wake-word detection system. In *Proc. Neural Information Processing Systems (NeurIPS)*, pages 11908–11918, 2019.

[31] Juncheng Li, Frank R. Schmidt, and J. Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *Proc. International Conference on Machine Learning, ICML*, volume 97, pages 3896–3904, 2019.

[32] Bo Luo, Yannan Liu, Lingxiao Wei, and Qiang Xu. Towards imperceptible and robust adversarial example attacks against neural networks. In *Thirty-second aaai conference on artificial intelligence*, 2018.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[34] Apostolos Modas, Rahul Rade, Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Prime: A few primitives can boost robustness to common corruptions. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV*, pages 623–640. Springer, 2022.

[35] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[36] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[37] Konda Reddy Mopuri, Aditya Ganeshan, and R Venkatesh Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2452–2465, 2018.

[38] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5636–5643, 2020.

[39] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[43] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *CoRR*, abs/2002.08347, 2020.

[44] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Enhancing gradient-based attacks with symbolic intervals, 2019.

[45] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

[46] Chaowei Xiao, Bo Li, Jun yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 3905–3911, 2018.

[47] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

[48] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In So Kweon. Understanding adversarial examples from the mutual influence of images and perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14521–14530, 2020.

[49] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In-So Kweon. Cd-uap: Class discriminative universal adversarial perturbation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6754–6761, 2020.

[50] Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally adversarial attack. *Proc. AAAI Conference on Artificial Intelligence*, 33:2253–2260, 07 2019.

Appendix

A Definitions

In this section, we will formally define the terms used in the main body of the paper. We first start with adversarial examples.

Definition A.1. *Given a correctly classified point* $\mathbf{x}$*, a distance function* $d(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$*, and bound* $\epsilon \in \mathbb{R}$*,* $\mathbf{x}'$ *is an adversarial example iff* $d(\mathbf{x}', \mathbf{x}) < \epsilon$ *and* $\hat{f}(\mathbf{x}') \neq y$*.*

We distinguish between adversarial examples and perturbations. An *adversarial perturbation* added to the point it is attacking is an *adversarial example*, $x' = x + v$. We can construct v by solving the following optimization problem:

$$\arg\min_{v} ||v||_p \text{ s.t. } \hat{f}(x+v) \neq \hat{f}(x) \tag{6}$$

Here, we are looking for the smallest v such that f's classification changes from the original output (assuming f correctly classified x).

Next, we define the adversarial success rate which measures whether or not a given perturbation is adversarial.

Definition A.2. *Given a datapoint* x*, and perturbation* v*, adversarial success, AS, is defined as*

$$AS(f, x, v) = 1 - \delta(\hat{f}(x+v), \hat{f}(x)) \tag{7}$$

Here, $\delta(i, j)$ *refers to the Kronecker Delta function [2], formally,*

$$\delta(i,j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{8}$$

With AS, we measure whether v *changes* f*'s classification of* x*.*

Using the definision of adversarial success we can now define unniversal adversarial success rate.

Definition A.3. *Given a data distribution* μ*, and perturbation* $\mathbf{u}$*, universal adversarial success rate,* ASR_U*, for* $\mathbf{u}$*, is*

$$ASR_U(f, \mu, \mathbf{u}) = \underset{\mathbf{x}\sim\mu}{P}(\hat{f}(\mathbf{x}+\mathbf{u}) \neq \hat{f}(\mathbf{x})) \tag{9}$$

Using Definition A.3, we formally define a UAP.

Definition A.4. *A universal adversarial perturbation is a vector* $\mathbf{u} \in \mathbb{R}^d$ *which, when added to almost all datapoints in* μ *causes the classifier* f *to misclassify. Formally, given* γ*, a bound on universal ASR, and* l_p*-norm with corresponding bound* ϵ*,* $\mathbf{u}$ *is a UAP iff* $ASR_U(f, \mu, \mathbf{u}) > \gamma$ *and* $||\mathbf{u}||_p < \epsilon$*.*

Now, we move onto definitions pertaining to robust UAPs. We start by formally defining transformation sets and neighborhoods.

Definition A.5. *A transformation,* τ*, is a composition of bijective sub-differentiable transformation functions. A transformation set,* T*, is a set of distinct transformations. A point* $\mathbf{v}'$ *is in the neighborhood* $N_T(\mathbf{v})$*, of* $\mathbf{v}$*, if there is a transform in* T *that maps* $\mathbf{v}$ *to* $\mathbf{v}'$*. Formally,*

$$\mathbf{v}' \in N_T(\mathbf{v}) \iff \exists \tau \in T \text{ s.t. } \tau(\mathbf{v}) = \mathbf{v}' \tag{10}$$

Finally, we formally define robust universal adversarial success rate.

Definition A.6. *Given a data distribution* μ*, transformation set* T*, universal ASR level* γ*, bound* ϵ *on* l_p*-norm, and perturbation* $\mathbf{u_r}$*, robust universal adversarial success rate,* ASR_R*, is defined as,*

$$ASR_R(f, \mu, T, \gamma, \mathbf{u_r}) = \underset{\mathbf{u'_r}\sim N_T(\mathbf{u_r})}{P}(ASR_U(f, \mu, \mathbf{u'_r}) > \gamma \wedge ||\mathbf{u'_r}||_p < \epsilon) \tag{11}$$

B Proof of Theorem 4.1

This proof relies heavily on similar proofs provided by Chernoff [13] and by Alippi [4]. We refer to the reader to these texts for further details. In our proof, we show how to adapt universal ASR to these proofs.

Proof. The bound on n is derived via the Chernoff inequality applied to $\hat{p}_n(\gamma)$ and $\mathbb{E}[\hat{p}_n(\gamma)] = p(\gamma)$ [13, 4]. Equation 5 holds since computing universal ASR is Lebesgue measurable over the data distribution and since we assume $N_T(\mathbf{u_r})$ has a well defined PDF. □

C Semantic Transformations

In this section, we discuss the semantic transformations used in the paper. Brightness and contrast can be represented via *bias* (β) and *gain* ($\alpha > 0$) parameters respectively. Formally, if $\mathbf{x}$ is the original image, then the transformed image, $\mathbf{x}'$, can be represented as

$$\mathbf{x}' = \alpha\mathbf{x} + \beta \tag{12}$$

Rotation, scaling, shearing, and translation are all affine transformations acting on the coordinate system, c, of the images instead of the pixel values, $\mathbf{x}$. In order to recover the pixel values and differentiate over the transformation, we will need sub-differentiable interpolation, see Appendix D. For finite dimensions, affine transformations can be represented as a linear coordinate map where the original coordinates are multiplied by an invertible augmented matrix and then translated with additional bias vector. Below, we give the general form for an affine transformation given augmented matrix $\mathbf{A}$, bias matrix $\mathbf{b}$, and input coordinates c. We can compute the output coordinates, c', as

$$\begin{bmatrix}\mathbf{c}' \\ 1\end{bmatrix} = \begin{bmatrix}[ccc|c] & \mathbf{A} & & \mathbf{b} \\ 0 & \dots & 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{c} \\ 1\end{bmatrix} \tag{13}$$

Below, we give the augmented matrix $\mathbf{A}$ and additional bias matrix $\mathbf{b}$ for rotation, scaling, shearing, and translation.

Rotation, $R(\theta)$, by θ degrees:

$$\mathbf{A} = \begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix}, \mathbf{b} = \begin{pmatrix}0 \\ 0\end{pmatrix} \tag{14}$$

Scaling, $Sc(p)$, by $p\%$:

$$\mathbf{A} = \begin{pmatrix}1+\frac{p}{100} & 0 \\ 0 & 1+\frac{p}{100}\end{pmatrix}, \mathbf{b} = \begin{pmatrix}0 \\ 0\end{pmatrix} \tag{15}$$

Shearing, $Sh(m)$, by shear factor $m\%$:

$$\mathbf{A} = \begin{pmatrix}1 & 1+\frac{m}{100} \\ 0 & 1\end{pmatrix}, \mathbf{b} = \begin{pmatrix}0 \\ 0\end{pmatrix} \tag{16}$$

Translation, $T(x, y)$, by x pixels horizontally and y pixels vertically:

$$\mathbf{A} = \begin{pmatrix}0 & 0 \\ 0 & 0\end{pmatrix}, \mathbf{b} = \begin{pmatrix}x \\ y\end{pmatrix} \tag{17}$$

D Interpolation

Affine transformations may change a pixel's integer coordinates into non-integer coordinates. Interpolation is typically used to ensure that the resulting image can be represented on a lattice (integer) pixel grid. For this paper, we will be using bilinear interpolation, a common interpolation method which achieves a good trade-off between accuracy and efficiency in practice and is commonly used in literature [47, 8]. Let $x_{i,j}$, $x'_{i,j}$ represent the pixel value at position i, j for the original and

transformed image respectively. Let ${c'}^x_{i,j}$, ${c'}^y_{i,j}$ represent the x-coordinate and y-coordinate of the pixel at i, j after transformation. We define our transformed image by summing over all pixels $n, m \in [1 \dots H] \times [1 \dots W]$ where H and W represent the height and width of the image.

$$x'_{i,j} = \sum_{n}^{H} \sum_{m}^{W} x_{n,m} \max(0, 1 - |{c'}^x_{i,j} - m|) \max(0, 1 - |{c'}^y_{i,j} - n|) \tag{18}$$

This interpolation can be computed for each channel in the image. While interpolation is typically not differentiable, in order to generate adversarial examples using standard techniques we need a differentiable version of interpolation. [26] introduces differentiable image sampling. Their method works for any interpolation method as long as the (sub-)gradients can be defined with respect to $x, {c'}_{i,j}$. For bilinear interpolation this becomes,

$$\frac{\partial x'_{i,j}}{\partial x_{n,m}} = \sum_{n}^{H} \sum_{m}^{W} \max(0, 1 - |{c'}^x_{i,j} - m|) \max(0, 1 - |{c'}^y_{i,j} - n|) \tag{19}$$

$$\frac{\partial x'_{i,j}}{\partial {c'}^x_{i,j}} = \sum_{n}^{H} \sum_{m}^{W} x_{n,m} \max(0, 1 - |{c'}^y_{i,j} - n|) \begin{cases} 1 & \text{if } m \geq |{c'}^x_{i,j} - m| \\ -1 & \text{if } m < |{c'}^x_{i,j} - m| \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

E Baseline Algorithms

E.1 Stochastic Gradient Descent

The first baseline directly solves Equation 2 using gradient descent. Since we are solving a constrained optimization problem, we cannot use gradient descent directly. Instead, we can solve the Lagrangian-relaxed form of the problem (adding a weighted norm term to the minimization problem), as in [12, 6], using a momentum based Stochastic Gradient Descent (SGD). Shafahi et al. [38] suggests that this is an effective method for generating standard UAPs. Our SGD algorithm is in Appendix F. In order to implement it, we replace the Delta function with a loss function, L. We iteratively converge towards the inner expectation by computing it in batches, and towards the outer expectation by sampling a large number of transformations. Given that we would like to estimate on a batch, $\hat{\mathbf{x}} \subset \mu$, and a random set of transformations sampled from T, $\hat{\tau} \subset T$, we can approximate using:

$$\frac{I(||\hat{\tau}_j(\mathbf{u_r})|| < \epsilon)}{|\hat{\mathbf{x}}| \times |\hat{\tau}|} \sum_{i=1}^{|\hat{\mathbf{x}}|} \sum_{j=1}^{|\hat{\tau}|} L[f(\hat{\mathbf{x}}_\mathbf{i} + \hat{\tau}_j(\mathbf{u_r})), f(\hat{\mathbf{x}}_\mathbf{i})] - \lambda ||\mathbf{u_r}||_p \tag{21}$$

E.2 Standard UAP Algorithm with Robust Adversarial Perturbations

For our second baseline, we leverage the standard UAP algorithm from Moosavi-Dezfooli et al. [36] (see Appendix G for the algorithm). The standard UAP algorithm takes each input, $\mathbf{x_i}$, computes the smallest additive change, $\Delta\mathbf{u}$, to the current perturbation, $\mathbf{u}$, that would make $\mathbf{u} + \Delta\mathbf{u}$ an adversarial perturbation for $\mathbf{x_i}$. Intuitively, over time the algorithm will approach a perturbation that works on most inputs in the training dataset. We modify this approach by computing robust adversarial perturbations rather than standard adversarial perturbations. At each point $\mathbf{x_i}$, we compute the smallest additive change, $\Delta\mathbf{u_r}$, to the current robust adversarial perturbation, $\mathbf{u_r}$, that would make $\mathbf{u_r} + \Delta\mathbf{u_r}$ a robust adversarial perturbation for $\mathbf{x_i}$. We search for robust adversarial perturbations by optimizing the expectation that a point in the neighborhood of $\mathbf{v_r}$ is adversarial while restricting the perturbation to an l_p norm of ϵ.

F SGD Algorithm

Our SGD UAP algorithm is based on standard momentum based SGD while optimizing over the objective proposed in 2, the algorithm details can be seen in Algorithm 2.

Algorithm 2 Stochastic Gradient Descent UAP Algorithm

Initialize $\mathbf{u_r} \leftarrow 0, \Delta\mathbf{u_r} \leftarrow 0$
repeat
  for $\mathbf{B} \in \mathbf{X}$ **do**
    Sample $\hat{t} \subset T$
    $\Delta\mathbf{u_r} \leftarrow \alpha\Delta\mathbf{u_r} - \frac{\nu}{|\hat{\mathbf{x}}|\times|\hat{t}|}\sum_{i=1}^{|\hat{\mathbf{x}}|}\sum_{j=1}^{|\hat{t}|}\nabla L[f(\hat{\mathbf{x}}_\mathbf{i} + \hat{t}_j(\mathbf{u_r})), f(\hat{\mathbf{x}}_\mathbf{i})]$
    Update the perturbation with projection:
    $\mathbf{u} \leftarrow \mathcal{P}_{p,\epsilon}(\mathbf{u_r} + \Delta\mathbf{u_r})$
  end for
until $ASR_R(f, \mathbf{X}, T, \gamma, \mathbf{u_r}) < \zeta$

G Iterative UAP Algorithm

Moosavi-Dezfooli et al. [36] introduces an iterative UAP algorithm, the algorithm can be seen in Algorithm 3.

Algorithm 3 Iterative Universal Perturbation Algorithm (Moosavi-Dezfooli et al. [36])

Initialize $\mathbf{u} \leftarrow 0$
repeat
  for $\mathbf{x_i} \in \mathbf{X}$ **do**
    if $\hat{f}(\mathbf{x_i} + \mathbf{u}) = \hat{f}(\mathbf{x_i})$ **then**
      Compute minimal adversarial perturbation:
      $\Delta\mathbf{u} \leftarrow \arg\min_\mathbf{r} ||\mathbf{r}||_2$ s.t. $\hat{f}(\mathbf{x_i} + \mathbf{u} + \mathbf{r}) \neq \hat{f}(\mathbf{x_i})$
      Update the perturbation with projection:
      $\mathbf{u} \leftarrow \mathcal{P}_{p,\epsilon}(\mathbf{u} + \Delta\mathbf{u})$
    end if
  end for
until $ASR_U(f, \mathbf{X}, \mathbf{u}) < \gamma$

H Estimate Robustness Algorithm

In this section, we give our algorithm for estimating the robustness of a UAP.

Algorithm 4 EstRobustness

Draw $n = \lceil \frac{1}{2\psi^2} \ln \frac{2}{\phi} \rceil$ samples $\tau_i \sim T$
Return $\hat{p}_n(\gamma) = \frac{1}{n}\sum_i^n I(ASR_U(f, \mathbf{X}, \tau_i(\mathbf{u_r})) > \gamma)$

I Experiment Parameters

In our experiments, we have capped all algorithms at 5 epochs or if they have achieved an ASR_R of 0.95. The UAPs are trained with the same transformation set that they are evaluated on. For algorithms running PGD internally, we have capped the number of iterations to 40.

J Further Evaluation of Uniform Noise

A table of results for uniform random noise can be seen in Table 4.

K UAP performance against semantic transformations on CIFAR-10

In this section, we show similar results as shown in ILSVRC-2012 but on CIFAR-10. Here, we again see that RobustUAP outperforms all other baselines.

Table 4: Robust ASR with uniform random noise, $\gamma = 0.8$.

Dataset	Transformation Set	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC 2012	$U(0.1)$	81.6%	94.2%	91.3%	**99.0%**
	$U(0.3)$	10.7%	68.9%	42.7%	**96.1%**
CIFAR-10	$U(0.1)$	66.0%	98.1%	96.1%	**100%**
	$U(0.3)$	5.8%	96.1%	47.6%	**100%**

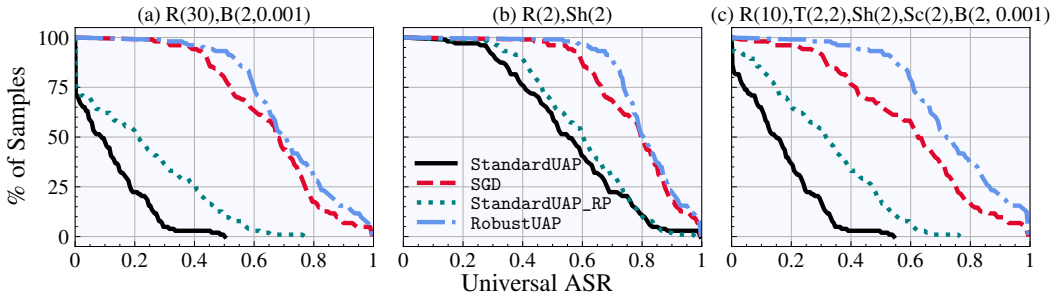

Figure 5: For each method, a point (x, y) in the corresponding line represents the percentage of sampled UAPs ($y\%$) with Universal ASR $> x$ for the different semantic transformations on CIFAR-10.

L Average ASR_U and ASR_R with different γ's

We provide additional metrics computed on the same set of transformations, datasets, and models as in Table 1. In Table 5, we present the Average ASR_U rather than ASR_R. The average shows us that our RobustUAP algorithm creates UAPs which after transformation on average are better UAPs than all other algorithms. We observe that the average shows us that even standard UAPs aren't completely ineffective after transformation they just have a very low chance of being highly effective.

Table 5: Average Universal ASR of our Robust UAP algorithms and the standard UAP [36] method.

Dataset	Transformation Set	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC 2012	$R(20)$	16.3%	71.5%	24.7%	**81.3%**
	$T(2,2)$	52.6%	82.6%	55.4%	**85.4%**
	$Sc(5), R(5), B(5, 0.01)$	44.9%	76.3%	58.5%	**82.2%**
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	13.6%	64.8%	29.0%	**75.3%**
CIFAR-10	$R(30), B(2, 0.001)$	9.9%	66.8%	22.2%	**73.4%**
	$R(2), Sh(2)$	57.1%	78.8%	61.2%	**82.9%**
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	16.2%	61.2%	32.6%	**76.4%**

In Table 6, we present ASR_R computed at $\gamma = [0.5, 0.7]$ rather than $\gamma = 0.6$. This table shows a similar story to above, and shows that our algorithm produces better results under a variety of success thresholds.

M Robustness to Random Noise

First, we show that our algorithm generates UAPs robust against uniform random noise. Here our neighborhood is defined as an L_∞ ball of radius ϵ around the perturbation. $U(\epsilon)$ represents noise drawn uniformly from such a ball. Figure 6 shows the performance of each algorithm. For example, the RobustUAP algorithm achieves a ASR_U of 0.9 greater than 97% of the time under $U(0.1)$ on CIFAR-10, where all other algorithms achieve 0.9 at most 30% of the time. RobustUAP outperforms

Table 6: Robust ASR of our Robust UAP algorithms and the standard UAP [36] method with $\gamma = [0.5, 0.7]$.

Dataset	Transformation Set	Standard UAP		SGD		Standard UAP_RP		Robust UAP	
		0.5	0.7	0.5	0.7	0.5	0.7	0.5	0.7
ILSVRC 2012	$R(20)$	1.9%	0.0%	88.3%	58.3%	10.7%	1.0%	**98.1%**	**76.7%**
	$T(2,2)$	51.5%	21.4%	**100%**	84.5%	57.3%	23.3%	**100%**	**91.3%**
	$Sc(5), R(5), B(5, 0.01)$	38.8%	11.7%	96.1%	67.0%	64.1%	25.2%	**99.0%**	**87.4%**
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	1.9%	0.0%	82.5%	38.8%	12.6%	1.0%	**95.1%**	**59.2%**
CIFAR-10	$R(30), B(2, 0.001)$	1.0%	0.0%	80.6%	43.7%	12.6%	1.0%	**93.2%**	**49.5%**
	$R(2), Sh(2)$	62.1%	22.3%	96.1%	68.9%	68.0%	30.1%	**99.0%**	**89.3%**
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	2.9%	0.0%	67.0%	38.8%	19.4%	1.0%	**93.2%**	**55.3%**

all other algorithms for both noise sizes. `StandardUAP` has a lower mean and higher variance in universal ASR and is much less robust to transformation. A table of Robust ASR results for $\gamma = 0.8$ can be seen in Appendix J. Our Robust ASR results are guaranteed to be ± 0.05 from the actual result with a probability of 95%. For example, we estimate that `RobustUAP` has ASR_R of 96.1% for U(0.3), we are guaranteed that the true robustness is $> 91.1\%$ with a probability of 95%. Note that we get robustness guarantees from `EstRobustness` as our neighborhood has a well-defined PDF.

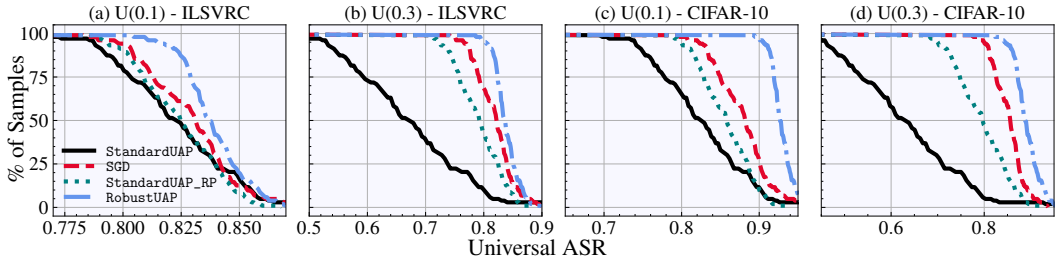

Figure 6: For each method, a point (x, y) in the corresponding line represents the percentage of sampled UAPs ($y\%$) with Universal ASR $> x$ for $U(0.1)$ and $U(0.3)$ on ILSVRC and CIFAR-10.

N Comparison on non-Robust Universal ASR metric

We compare our robust UAPs to standard UAPs on the non-robust universal ASR metric, see Table 7. All robust UAPs are generated to be robust against $R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$. We observe that at the same l_2-norm all robust UAPs achieve a lower universal ASR than the standard UAP algorithm. This result is not too surprising as solving the optimization problem for robust UAP is significantly more difficult. We further observe that our `RobustUAP` algorithm is the most effective in comparison to the other robust baseline approaches.

Table 7: Universal ASR of our Robust UAP algorithms and the standard UAP method.

Dataset	StandardUAP	SGD	StandardUAP_RP	RobustUAP
ILSVRC 2012	**95.5%**	85.6%	82.3%	91.3%
CIFAR-10	**96.2%**	89.3%	84.0%	93.7%

O Additional Models

We also provide additional data on our methods evaluated on the same transformations and datasets but on different models. In this case, we use ResNet-18 [22] for CIFAR-10 and MobileNet [25] for ILSVRC 2012. Results can be seen in Table 8. We observe similar performance across models suggesting that the performance of the attacks is more directly tied to transformation set and dataset.

Table 8: Robust ASR on Resnet-18 for CIFAR-10 and MobileNet for ILSVRC 2012.

Dataset	Model	Transformation Set	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC 2012	MobileNet	$R(20)$	8.1%	71.2%	2.6%	**85.0%**
		$T(2,2)$	40.9%	98.7%	54.3%	**99.6%**
		$Sc(5), R(5), B(5, 0.01)$	16.3%	94.5%	44.3%	**96.3%**
		$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	4.1%	75.7%	8.6%	**86.2%**
CIFAR-10	ResNet-18	$R(30), B(2, 0.001)$	0.9%	67.8%	6.4%	**74.9%**
		$R(2), Sh(2)$	49.9%	99.5%	49.1%	**99.8%**
		$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	8.0%	70.8%	12.2%	**83.8%**

P Common Corruptions

We also evaluate robust UAP against the 2D fog transformations in [27]. We set the shift intensity of the fog to be 1 and train our robust UAPs to be robust against random fog perturbations. We observe similar results to the transformations we experiment with above. The graph of the results can be seen in Figure 7.

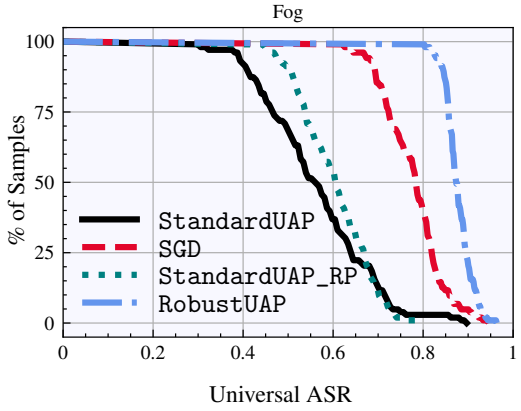

Figure 7: For each method, a point (x, y) in the corresponding line represents the percentage of sampled UAPs ($y\%$) with Universal ASR $> x$ for the different semantic transformations on ILSVRC-2012.

Q Algorithm Runtimes

We compare the average runtimes of the different methods on one of our most challenging $R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$ transformation set on ILSVRC-2012 and $n = 738$. The results are in Table 9. We observe that `RobustUAP` is the slowest algorithm and `SGD` is the fastest. `RobustUAP` uses `EstimateRobustness` in each loop and thus with high n it requires much more time to compute. The extra computation enables `Robust UAP` to obtain better robustness than all baselines. On the same set of transformations and dataset we observe that one iteration of `EstimateRobustness` on the entire test set takes on average 19 minutes. When running `EstimateRobustness` in the `RobustUAP` loop, each call takes 36 seconds for a batch size of 32.

Table 9: Average Runtime for Robust UAP algorithms

ALGORITHM	TIME(MIN)
STANDARD UAP	37
SGD	32
STANDARD UAP_RP	43
ROBUST UAP	118

R Effect of Compute Time on Robustness

Previous sections highlight SGD as the most competitive algorithm to RobustUAP in terms of performance. However, in the previous section we note that SGD takes significantly less time to run. In this section, we investigate how RobustUAP performs with limited compute time as well as how SGD performs with increased runtime. We first add results to the ILSVRC 2012 part of Table 1 by also computing RobustUAP performance when limited to the same amount of time that SGD takes. Table 10 shows that RobustUAP outperforms SGD even when its compute time is limited with up to 9% more robustness on our most challenging transformation $R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$.

Table 10: Robust ASR of RobustUAP restricted to the same amount of compute time as SGD.

DATASET	TRANSFORMATION SET	SGD	ROBUST UAP	RESTRICTED ROBUST UAP
ILSVRC 2012	$R(20)$	69.9%	**93.2%**	72.9%
	$T(2,2)$	96.1%	**97.1%**	96.9%
	$Sc(5), R(5), B(5, 0.01)$	85.4%	**96.1%**	86.3%
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	63.1%	**86.4%**	72.0%

Next, we vary the number of SGD iterations. We compute the robust ASR on ILSVRC for robustness against $R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$. Figure 8, shows the robust ASR achieved by SGD over time, here we observe that SGD's performance flatlines after a small number of iterations and seems to be unable to surpass about 65. Here SGD is allowed to continue to run past where it would usually stop (at around 250 iterations), in this experiment we allow it to go to 1250 iterations which is about the same amount of time that RobustUAP takes to run. RobustUAP is able to achieve a performance of 72 even when restricted to the amount of compute time of base SGD (It achieves 86.4 when unrestricted). These two results in combination show that RobustUAP is able to find more robust UAPs than SGD whose performance stabilizes.

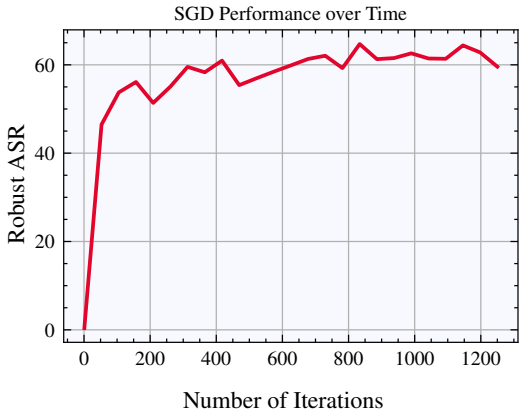

Figure 8: The Robust ASR with $\gamma = 0.6$ for SGD over time

S Targeted Attack

So far in this paper we have focused on untargeted attacks, i.e. attacks which aim to degrade the general performance of the model. Targeted attacks are also possible with both standard adversarial attack methods and universal adversarial perturbation methods. Here, we can simply turn our algorithm from untargeted to targeted by replacing the loss function. We would like to have target class, A, be classified as target class, B. Instead of maximizing the expected value of the cross entropy loss we can instead formulate the loss based on maximizing B while minimizing A similar to [9]. For ILSVRC 2012, we randomly select a couple of target classes and perform this attack, for each of these cases, we train our robust UAP to be robust to $R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$. Table 11 shows our results for robust ASR with $\gamma = 0.6$. We are measuring our robust ASR of turning class A into class B and observe similar results with `RobustUAP` being the most robust followed by `SGD`. It is also interesting to note that different random combinations lead to more or less success, i.e. it is easier to turn a dog into another dog than perfume into a padlock.

Table 11: Robust ASR of `RobustUAP` for target to target attack compared to the three baselines with $\gamma = 0.6$.

Dataset	Target Class	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC-2012	toy poodle → maltese dog	42.4%	99.1%	85.6%	**99.8%**
	perfume → padlock	0.0%	63.8%	5.1%	**76.4%**

T `RobustUAP` vs `SGD` Performance.

When comparing our baselines to `RobustUAP` we observe that the `SGD` algorithm has the most comparable performance to `RobustUAP`. In Appendix Q, we report the runtimes for the different algorithms and observe that `SGD` runs four times as fast as `RobustUAP`. Although this seems to suggest that `SGD` is more efficient, we further investigate restricting the runtime of `RobustUAP` in Appendix R. We find that `RobustUAP` beats out `SGD` at the same runtime. Furthermore, we observe that `SGD`'s performance flatlines and does not reach the performance of `RobustUAP` even when allowed to run for longer. Finally, we note that since UAPs only need to be computed a single time and can be done in advance, the runtime considerations are not a big factor for most practical use cases.

U Data Efficiency

In this section, we will evaluate the data efficiency of `RobustUAP`. We use `RobustUAP` to generate UAPs robust to $R(10), T(2,2), Sh(2), Sc(2), B(2,0.001)$ on ILSVRC-2012 with differing amounts of training data. The results can be seen in Figure 9. These results show that the algorithm is able to achieve good performance at 500 data points but continues to improve up to 4000 data points. After that it seems to stagnate.

V Transformer-based Models

Recently, transformers have become popular as a new architecture for deep learning models for computer vision tasks. In this section, we evaluate the effectiveness of robust UAPs against one such model, ViT [17]. Benz et al. [10] has shown that standard UAPs are still effective against transformer based architectures. In Table 12 we can see that we get similar results compared to our results on Inception and MobileNet. This shows that our methods work against transformer based models as well.

W Robust UAPs against Robustly Trained Networks

In this section, we are interested in seeing whether training networks to be robust against the same transformations that the UAP is trying to be robust against is helpful. For this, we trained two new

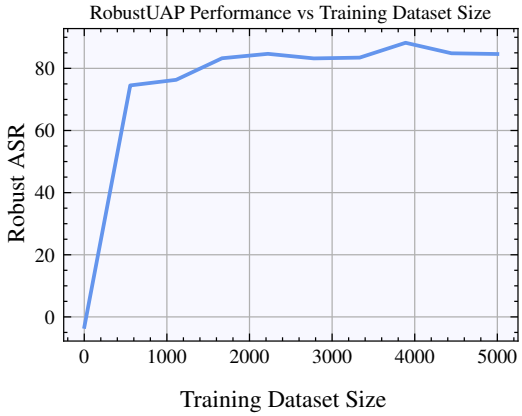

Figure 9: Robust ASR with $\gamma = 0.6$ for `RobustUAP` with differing amounts of training data

Table 12: Robust ASR of `RobustUAP`compared to the three baselines for ViT.

Dataset	Model	Transformation Set	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC-2012	ViT	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	2.0%	72.1%	12.9%	**88.5**%

Inception-v3 networks. Because of time limitations, we started with our base Inception-v3 network and fine-tuned it using data augmentations. For the first network InceptionR20, we augmented the data by adding random rotations within 20 degrees. For the second network InceptionHF, we augmented the data by adding horizontal flips. We then crafted UAPs robust against rotations and flips on InceptionR20 and InceptionHF respectively. The results can be seen in Table 13. We can compare the $R(20)$ results to those from our normal inception network. We postulate that since the network has received some additional robustness training it is harder to attack, and thus we should see slightly lower robustness scores. However, it seems that training the network to be robust to $R(20)$ does not significantly effect the ability to create robust UAPs. The horizontal flips seems like it might be too easy of a transformation as even standard UAP performs quite well for robust ASR.

Table 13: Robust ASR of `RobustUAP` compared to the three baselines for robust networks.

Dataset	Model	Transformation Set	Standard UAP	SGD	Standard UAP_RP	Robust UAP
ILSVRC-2012	InceptionR20	$R(20)$	6.3%	72.4%	10.2%	**81.3**%
	InceptionHF	HF	81.3%	99.5%	89.7%	**99.6**%

X Ablation on optimization strategy

In this section, we study the effect of using different optimizers in addition to SGD. We use a variety of standard PyTorch optimizers, Adam, Adamax, Adagrad, and RMSProp. We formulate the optimization problem in the same way but instead use these algorithms in order to optimize our perturbation. We compute these results on ILSVRC-2012 with Inception-v3 and use $R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$ as the transformation set and with $\gamma = 0.6$. The results can be seen in Table 14. We see that the optimization strategy has some affect on the results and that SGD performs the best. We also found that SGD performed marginally faster than the rest of the approaches.

Table 14: Comparison of different optimization strategies.

OPTIMIZER	ASR_R
SGD	**63.1**%
ADAM	59.7%
ADAMAX	60.1%
ADAGRAD	62.3%
RMSPROP	58.3%

Y Visualization of Robust UAPs generated with Different Algorithms

We further visualize UAPs generated with our three robust algorithms on the same transformation set against a standard UAP generated on ILSVRC 2012 in Figure 10. We observe that `StandardUAP` UAPs resemble `StandardUAP_RP` UAPs, but `StandardUAP_RP` algorithm concentrates its budget towards the center as the center is least likely to be perturbed under our transformation set. Both the `RobustUAP` and the `SGD` algorithm generate larger patterns across the image.

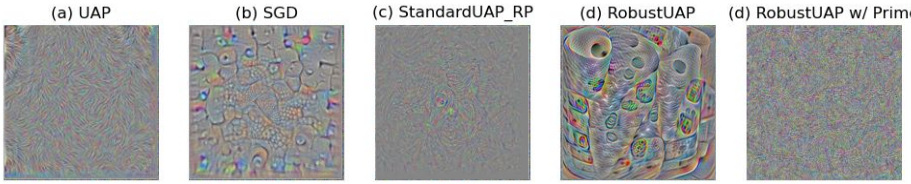

Figure 10: Comparison of UAPs on ILSVRC 2012 generated with (a) `StandardUAP`, (b) `RobustUAP`, (c) `Standard UAP_RP`, (d) `RobustUAP`, and (e) `RobustUAP` generated on Prime.

Z Error Bars

In this section, we will provide error bars/standard deviations for results reported in the paper.

Z.1 Table 1

First, we report the standard deviations for Table 1 which we obtain by learning each UAP 10 times then evaluating them on each their respective dataset/transformation set combinations.

Table 15: Standard Deviation of Robust ASR values reported in Table 1

DATASET	TRANSFORMATION SET	STANDARD UAP	SGD	STANDARD UAP_RP	ROBUST UAP
ILSVRC 2012	$R(20)$	1.1%	7.2%	10.2%	1.5%
	$T(2,2)$	11.4%	1.8%	6.3%	2.3%
	$Sc(5), R(5), B(5, 0.01)$	10.1%	5.2%	7.4%	3.1%
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	0.0%	8.6%	4.9%	2.8%
CIFAR-10	$R(30), B(2, 0.001)$	0.2%	7.5%	9.0%	5.2%
	$R(2), Sh(2)$	11.3%	5.5%	8.2%	0.9%
	$R(10), T(2,2), Sh(2), Sc(2), B(2, 0.001)$	1.8%	6.8%	5.1%	3.7%

Z.2 Table 2

First, we report the standard deviations for Table 2 which we obtain by learning each UAP 10 times then evaluating them on each their respective dataset/transformation set combinations.

Table 16: Robust ASR (%) of `RobustUAP` trained on PRIME, Affine ($R(10)$, $T(2,2)$, $Sh(2)$, $Sc(2)$, $B(2, 0.001)$), and Fog when applied to Prime, Affine, and common corruption transforms

	Evaluation Corruption Set																
Train Set			Noise			Blur				Weather				Digital			
	Prime	Aff.	Gaus.	Shot	Imp.	Defo.	Glass	Moti.	Zoom	Snow	Fog	Frost	Bright	Contr.	Elast.	Pixel	JPEG
PRIME	3.7	4.6	6.1	3.2	4.6	1.3	3.8	4.9	6.7	2.4	7.5	4.3	1.4	5.2	3.1	4.6	2.7
Affine	9.7	5.9	3.0	2.7	5.2	8.7	7.6	5.2	0.7	11.2	6.9	6.6	5.2	0.6	3.8	9.6	6.8
Fog	3.1	5.2	4.7	6.6	4.6	1.9	8.4	3.5	5.1	17.8	1.6	12.1	4.0	3.6	7.3	1.1	12.6

Z.3 Remaining Values

We find that the standard deviations are pretty similar across both tables reported and in some testing of the remaining results. For time reasons we have left the remaining standard deviations out as we don't find them informative. We are happy to provide these numbers for any results in the main body or appendix of the paper.