Ask, Acquire, and Attack: Data-free UAP Generation using Class Impressions

Konda Reddy Mopuri*[0000−0001−8894−7212], Phani Krishna Uppala*[0000−0003−0413−5685], and R. Venkatesh Babu[0000−0002−1926−1804]

Video Analytics Lab, Indian Institute of Science, Bangalore, India

Abstract. Deep learning models are susceptible to input specific noise, called adversarial perturbations. Moreover, there exist input-agnostic noise, called Universal Adversarial Perturbations (UAP) that can affect inference of the models over most input samples. Given a model, there exist broadly two approaches to craft UAPs: (i) data-driven: that require data, and (ii) data-free: that do not require data samples. Data-driven approaches require actual samples from the underlying data distribution and craft UAPs with high success (fooling) rate. However, data-free approaches craft UAPs without utilizing any data samples and therefore result in lesser success rates. In this paper, for data-free scenarios, we propose a novel approach that emulates the effect of data samples with class impressions in order to craft UAPs using data-driven objectives. Class impression for a given pair of category and model is a generic representation (in the input space) of the samples belonging to that category. Further, we present a neural network based generative model that utilizes the acquired class impressions to learn crafting UAPs. Experimental evaluation demonstrates that the learned generative model, (i) readily crafts UAPs via simple feed-forwarding through neural network layers, and (ii) achieves state-of-the-art success rates for data-free scenario and closer to that for data-driven setting without actually utilizing any data samples.

Keywords: adversarial attacks · attacks on ML systems · data-free attacks · image-agnostic perturbations · class impressions

1 Introduction

Machine learning models are pregnable (e.g. [4,3,9]) at test time to specially learned, mild noise in the input space, commonly known as adversarial perturbations. Data samples created via adding these perturbations to clean samples are known as adversarial samples. Lately, the Deep Neural Networks (DNN) based object classifiers are also observed [28,7,14,11] to be drastically affected by the adversarial attacks with quasi imperceptible perturbations. Further, it is observed (e.g. [28]) that these adversarial perturbations exhibit cross model generalizability (transferability). This means, often same adversarial sample gets

*Equal contribution

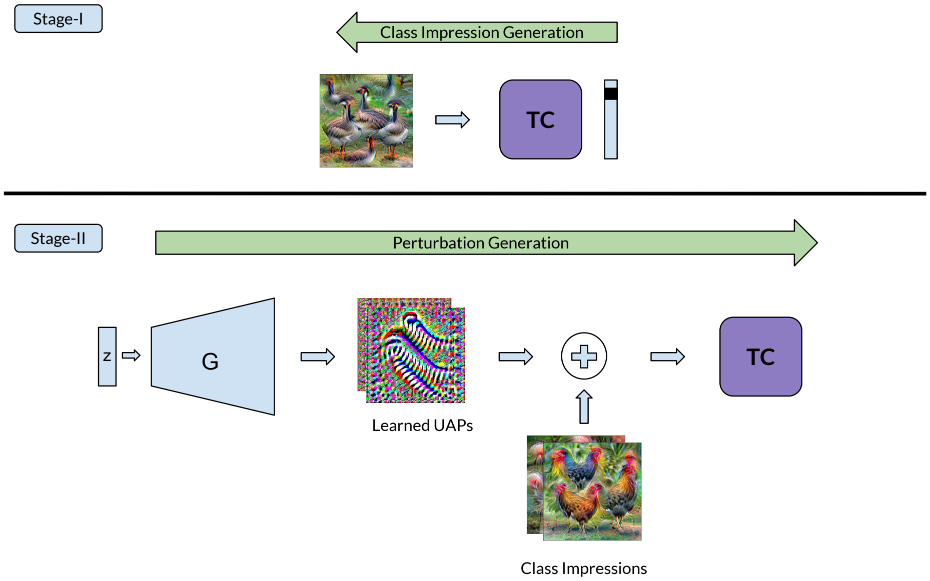


Fig. 1. Overview of the proposed approach. Stage-I, "Ask and Acquire" generates the "class impressions" to mimic the effect of actual data samples. Stage-II, "Attack" learns a neural network based generative model G which crafts UAPs from random vectors z sampled from a latent space.

incorrectly classified by multiple models in spite of having different architectures and trained with disjoint training datasets. It enables attackers to launch simple black-box attacks [21,12] on the deployed models without any knowledge about their architecture and parameters.

However, most of the existing works (e.g. [28,14]) craft input-specific perturbations, i.e., perturbations are functions of input and they may not transfer across data samples. In other words, perturbation crafted for one data sample most often fails to fool the model when used to corrupt other clean data samples. However, recent findings by Moosavi-Dezfooli *et al.* [13] and Mopuri *et al.* [17,15] demonstrated that there exist input-agnostic (or image-agnostic) perturbations that when added, most of the data samples can fool the target classifier. Such perturbations are known as "Universal Adversarial Perturbations (UAP)", since a single noise can adversarially perturb samples from multiple categories. Furthermore, it is observed that similar to image-specific perturbations, UAPs also exhibit cross model generalizability enabling easy black-box attacks. Thus, UAPs pose a severe threat to the deployment of the vision models and require a meticulous study. Especially for applications which involve safety (e.g. autonomous driving) and privacy of the users (e.g. access granting), it is indispensable to develop robust models against such adversarial attacks.

Approaches that craft UAPs can be broadly categorized into two classes: (i) data-driven, and (ii) data-free approaches. Data-driven approaches such as [13] require access to samples of the underlying data distribution to craft UAPs using a fooling objective (e.g. confidence reduction as in eq (2)). Thus, UAPs crafted via data-driven approaches typically result in higher success rate (or fooling rate), i.e., fool the models more often. Note that data-driven approaches have access to the data samples and the model architecture along with the parameters. Further, performance of the crafted UAPs is observed ([17,15]) to be proportional to the number of data samples available during crafting. However the data-free approaches (e.g. FFF [17]), with a goal to understand the true stability of the the models, indirectly craft UAPs (e.g. activation loss of FFF [17]) instead of using a direct fooling objective. Note that data-free approaches have access to only the model architecture and parameters but not to any data samples. Thus, it is a challenging problem to craft UAPs in data-free scenarios and therefore the success rate of these UAPs would typically be lesser compared to that achieved by the data-driven ones.

In spite of being difficult, data-free approaches have important advantages:

- When compared to their data-driven counter parts, data-free approaches reveal accurate vulnerability of the learned representations and in turn the models. On the other hand, success rates reported by data-driven approaches act as a sort of upper bounds on the achievable rates. Also, it is observed ([17,15]) that their performance is proportional to the amount of data available for crafting UAPs.
- Because of the strong association of the data-driven UAPs to the target data, they suffer poor transferability across datasets. On the other hand, data-free UAPs transfer better across datasets [17,15].
- Data-free approaches are typically faster [17] to craft UAPs.

Thus, in this paper, we attempt to achieve best of both worlds, i.e., effectiveness of the data-driven objectives and efficiency, transferability of the data-free approaches. We present a novel approach for the data-free scenarios that emulates the effect of actual data samples with "*class impressions*" of the model and crafts UAPs via learning a feed-forward neural network. Class impressions are the reconstructed images from the model's memory which is the set of learned parameters. In other words, they are generic representations of the object categories in the input space (as shown in Fig. 2). In the first part of our approach, we acquire class impressions via simple optimization (sec. 3.2) that can serve as representative samples from the underlying data distribution. After acquiring multiple class impressions for each of the categories, we perform the second part, which is learning a generative model (a feed-forward neural network) for efficiently generating UAPs. Thus, unlike the existing works ([13,17]) that solve complex optimizations to generate UAPs, our approach crafts via a simple feed-forward operation through the learned neural network. The major contributions of our work can be listed as:

- We propose a novel approach to handle the absence of data (via class impressions, sec. 3.2) for crafting UAPs and achieve state-of-the-art success (fooling) rates.
- We present a generative network (sec. 3.3) that learns to efficiently generate UAPs utilizing the class impressions.

The paper is organized as followed: section 2 describes the relevant existing works, section 3 presents the proposed framework in detail, section 4 reports comprehensive experimental evaluation of our approach and finally section 5 concludes the paper.

2 Related Works

Adversarial perturbations (e.g. [28,7,14]) reveal the vulnerability of the learning models to specific noise. Further, these perturbations can be input agnostic [13,17] called "Universal Adversarial Perturbations (UAP)" and can pose severe threat to the deployability of these models. Existing approaches to craft the UAPs ([13,17,15]) perform complex optimizations every time we wish to craft a UAP. Differing from the previous works, we present a neural network that readily crafts UAPs. Only similar work by Baluja *et al.* [2] presents a neural network that transforms a clean image into an adversarial sample by passing through a series of layers. However, we learn a generative model which maps a latent space to that of UAPs. A concurrent work by Mopuri *et al.* [18] presents a similar generative model approach to craft perturbations but for data-driven case.

Also, existing data-free method [17] to craft UAPs achieves significantly less success rates compared to the data-driven methods such as UAP [13] and NAG [18]. In this paper, we attempted to reduce the gap between them by emulating the effect of data with the proposed class impressions. Our class impressions are obtained via simple optimization similar to visualization works such as [26,27]. Feature visualizations [26,27,29,31,25,30,16] are introduced (i) to understand what input patterns each neuron responds to, and (ii) gain intuitions into neural networks in order to alleviate the black-box nature of the neural networks. Two slightly different approaches exist for feature visualizations. In the first approach, a random input is optimized in order to maximize the activation of a chosen neuron (or set of neurons) in the architecture. This enables to generate visializations for a given neuron (as in [26]) in the input space.

In other approaches such as the Deep Dream [19] instead of choosing a neuron to activate, arbitrary natural image is passed as an input, and the network enhances the activations that are detected. This way of visualization finds the subtle patterns in the input and amplify them. Since our task is to generate class impressions that emulate the behaviour of real samples, we follow the former approach.

Since the objective is to generate class impressions that can be used to craft UAPs with the fooling objective, softmax probability neuron seems like the obvious choice to activate. However, this intuition is misleading, [26,20] have shown that directly optimizing at softmax leads to increase in the class probability by reducing the pre-softmax logits of other classes. Also, often it does not increase the pre-softmax value of the desired class, thus giving poor visualizations. In order to make the desired class more likely, we optimize the pre-softmax logits and our observations are in agreement with that of [26,20].

3 Proposed Approach

In this section we present the proposed approach to craft efficient UAPs for data-free scenarios. It is understood ([13,17,18]) that, because of data availability and a more direct optimization, data-driven approaches can craft UAPs that are effective in fooling. On the other hand, the data-free approaches can quickly craft generalizable UAPs by solving relatively simple and indirect optimizations. In this paper we aim to achieve the effectiveness of the data-driven approaches in the data-free setup. For this, first we create representative data samples called, *class impressions* (Figure 2) to mimic the actual data samples of the underlying distribution. Later, we learn a neural network based generative model to craft UAPs using the generated class impressions and a direct fooling objective (eq.(2)). Figure 1 shows the overview of our approach. Stage-I, “Ask and Acquire” is about the class impression generation from the target CNN model and Stage-II, “Attack” is training the generative model that learns to craft UAPs using the class impressions obtained in the first stage. In the following subsections, we will discuss these two stages in detail.

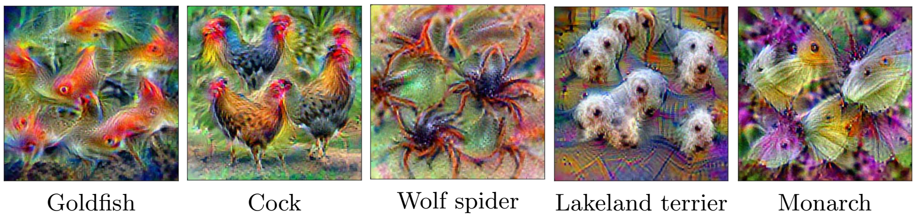
3.1 Notation

We first define the notations followed throughout this paper:

- f: target classifier (TC) under attack, which is a trained model with frozen parameters
- f^i_k: k^{th} activation in i^{th} layer of the target classifier
- $f^{ps/m}$: output of the pre-softmax layer
- $f^{s/m}$: output of the softmax (probability) layer
- v: additive universal adversarial perturbation (UAP)
- x: clean input to the target classifier, typically either data sample or class impression
- ξ: max-norm (l_1) constraint on the UAPs, i.e., maximum allowed strength of perturbation that can be added or subtracted at each pixel in the image

3.2 Ask and Acquire the Class Impressions

Availability of the actual data samples can enable to solve for a direct fooling objective thus craft UAPs that can achieve high success rates [13]. Hence in the data-free scenarios we generate samples that act as proxy for data. Note that the attacker has access to only the model architecture and the learned parameters



Goldfish Cock Wolf spider Lakeland terrier Monarch

Fig. 2. Sample class impressions generated for VGG-F [5] model. The name of the corresponding categories are mentioned below the images. Note that the impressions have several natural looking patterns located in various spatial locations and in multiple orientations.

of the target classifier (CNN). The learned parameters are a function of training data and procedure. They can be treated as model's memory in which the essence of training has been encoded and saved. The objective of our first stage, "Ask and Acquire" is to tap the model's memory and acquire representative samples of the training data. We can then use only these representative samples to craft UAPs to fool the target classifier.

Note that we do not aim to generate natural looking data samples. Instead, our approach creates samples for which the target classifier predicts strong confidence. That is, we create samples such that the target classifier strongly believes them to be actual samples that belong to categories in the underlying data distribution. In other words, these are impressions of the actual training data that we try to reconstruct from model's memory. Therefore we name them *Class Impressions*. The motivation to generate these class impression is that, for the purpose of optimizing a fooling objective (e.g. eq. 2) it is sufficient to have samples that behave like natural data samples, which is, to be predicted with high confidence. Thus, the ability of the learned UAPs to act as adversarial noise to these samples with respect to the target classifier generalizes to the actual samples.

Top panel of Fig. 1 shows the first stage of our approach to generate the class impressions. We begin with a random noisy image sampled from $\mathcal{U}[0, 255]$ and update it till the target classifier predicts a chosen category with high confidence. We achieve this via performing the optimization shown in eq (1). Note that we can create impression (CI_c) for any chosen class (c) by maximizing the predicted confidence to that class. In other words, we modify the random (noisy) image till the target network believes it to be an input from a chosen class c with high confidence. We consider the activations in the pre-softmax layer $f_c^{ps/m}$ (before we apply the softmax non-linearity) and maximize the model's confidence.

$$CI_c = \underset{x}{\text{argmax}} \quad f_c^{ps/m}(x) \tag{1}$$

While learning the class impressions, we perform typical data augmentations such as (i) random rotation in $[-5^o, 5^o]$, (ii) scaling by a factor randomly selected from $\{0.95, 0.975, 1.0, 1.025\}$, (iii) RGB jittering, and (iv) random cropping. Along with the above typical augmentations, we also add random uniform noise in $\mathcal{U}[-10, 10]$. Purpose of this augmentation is to generate robust impressions that behave similar to natural samples with respect to the augmentations and random noise. We can generate multiple impressions for a single category by varying the initialization, i.e., multiple initializations result in multiple class impressions. Note that the dimensions of the generated impressions would be same as that required by the model's input (e.g., $224 \times 224 \times 3$). We have implemented the optimization given in eq (1) in TensorFlow [1] framework. We used Adam [10] optimizer with a learning rate of 0.1 with other parameters set to their default values. In order to mimic the variety in terms of the difficulty of recognition (from easy to difficult samples), we have devised a stopping criterion for the optimization. We presume that the difficulty is inversely related to the confidence predicted by the classifier. Before we start the optimization in eq. (1), we randomly sample a confidence value uniformly in $[0.55, 0.99]$ range and stop our optimization after the predicted confidence by the target classifier reaches that. Thus, the generated class impressions will have samples of varied difficulty.

Fig. 2 shows sample class impressions generated for VGG-F [5] model. The corresponding category labels are mentioned below the impressions. Note that the generated class impressions clearly show several natural looking patterns located in various spatial locations and in multiple orientations. Fig. 3 shows multiple class impressions generated by our method starting from different initializations for "Squirrel Monkey" category. Note that the impressions have different visual patterns relevant to the chosen category. We have generated 10 class impressions for each of the 1000 categories in ILSVRC dataset resulting in a total of 10000 class impressions. These samples will be used to learn a neural network based generative model that can craft UAPs through a feed-forward operation.

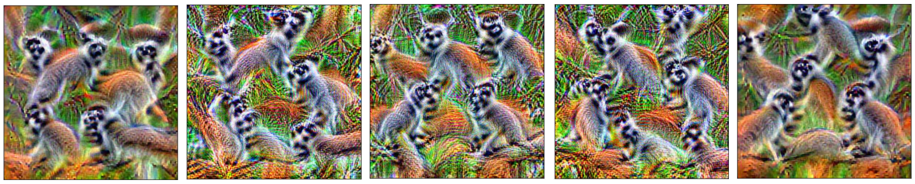


Fig. 3. Multiple class impressions for "Squirrel Monkey" category generated from different initializations for VGG-F [5] target classifier.

3.3 Attack: Craft the data-free perturbations

After generating the class impressions in the first stage of our approach, we treat them as training data for learning a generator to craft the UAPs. Bottom panel of Fig. 1 shows the overview of our generative model. In the following subsections we present the architecture of our model along with the objectives that drive the learning.

3.4 Fooling loss

We learn a neural network (G) similar to the generator part of a Generative Adversarial network (GAN) [6]. G takes a random vector z whose components are sampled from a simple distribution (e.g. $\mathcal{U}[-1,1]$) and transforms it into a UAP via a series of deconvolution layers. Note that in practice a mini-batch of vectors is processed. We train G in order to be able to generate the UAPs that can fool the target classifier over the underlying data distribution. To be specific, we train with a fooling loss computed over the generated class impressions (from Stage-I, sec. 3.2) as the training data. Let us denote the predicted label on clean sample (x) as 'clean label' and that of a perturbed sample ($x+v$) as 'perturbed label'. The objective is to make the 'clean' and 'perturbed' labels different. To ensure this to happen, our training loss reduces the confidence predicted to the 'clean label' on the perturbed sample. Because of the softmax nonlinearity, confidence predicted to some other label increases and eventually causes a label flip, which is fooling the target classifier. Hence, we formulate our fooling loss as

$$L_f = -log(1 - f_c^{s/m}(x+v)) \tag{2}$$

where c is the clean label predicted on x and $f_c^{s/m}$ is the probability (softmax output) predicted to category c. Note that this objective is similar to most of the adversarial attacking methods (e.g. FGSM [7,21]) in spirit.

3.5 Diversity loss

Fooling loss L_f (eq.(2)) only trains G to learn UAPs that can fool the target classifier. In order to avoid learning a degenerate G which can only generate a single strong UAP, we enforce diversity in the generated UAPs. We enforce that the crafted UAPs within a mini-batch are diverse via maximizing the pairwise distance between their embeddings $f^l(x+v_i)$ and $f^l(x+v_j)$, where v_i and v_j belong to generations within a mini-batch. We consider the layers of the target CNN for projecting (x+v). Thus our training objective is comprised of a diversity loss given by

$$L_d = -\sum_{i,j=1, i \neq j}^{K} d(f^l(x+v_i), f^l(x+v_j)) \tag{3}$$

where K is the mini-batch size, and d is a suitable distance metric (e.g., Euclidean or cosine distance) computed between the features extracted between a pair of adversarial samples. Note that the class impression x present in the two embeddings $f(x+v_i)$and$f(x+v_j)$ is same. Therefore, pushing them apart via minimizing L_d will make the UAPs v_i and v_j dissimilar.

Therefore the loss we optimize for training our generative model for crafting UAPs is given by

$$Loss = L_f + \lambda L_d \tag{4}$$

Note that this objective is similar in spirit to that presented in the concurrent work [18].

4 Experiments

In this section we present our experimental setup and the effectiveness of the proposed method in terms the success rates achieved by the crafted UAPs. For all our experiments we have considered ILSVRC [23] dataset and recognition models trained on it as the target CNNs. Note that, since we have considered data-free scenario, we extract class impressions to serve as data samples. Similar to the existing data-driven approach ([13]) that uses 10 data samples per class, we also extract 10 impressions for each class which makes a training data of 10000 samples.

4.1 Implementation details

The dimension of the latent space is chosen as 10, i.e, z is random $10D$ vector sampled from $\mathcal{U}[-1,1]$. We have investigated with other dimensions (e.g. 50, 100, etc.) for the latent space and found that 10 is efficient with respect to the number of parameters though the success rates are not very different. We used a mini-batch size of 32. All our experiments are implemented in TensorFlow [1] using Adam optimizer and the implementations are made available at https://github.com/val-iisc/aaa. The generator part (G) of the network maps the latent space Z to the UAPs for a given target classifier. The architecture of our generator consists of 5 deconv layers. The final deconv layer is followed by a $tanh$ non-linearity and scaling by ξ. Doing so limits the perturbations to $\left[-\xi, \xi\right]$. Similar to [13,17], the value of ξ is chosen to be 10 in order to add negligible adversarial noise. The architecture of G is adapted from [24]. We experimented on a variety of CNN architectures trained to perform object recognition on the ILSVRC [23] dataset. The generator (G) architecture is unchanged for different target CNN architectures and separately learned with the corresponding class impressions.

While computing the diversity loss (eq. 3), for each of the class impressions in the mini-batch (x), we select a pair of generated UAPs (v_1 and v_2) and compute the distance between $f^l(x+v_1)$ and $f^l(x+v_2)$. The diversity loss would be sum of all such distances computed over the mini-batch members. We typically consider the softmax layer of the target CNN for extracting the embeddings. Also, since the embeddings are probability vectors, we use cosine distance between the extracted embeddings. Note that, we can use any other intermediate layer for embedding and Euclidean distance for measuring their separation.

Since our objective is to generate diverse UAPs that can fool effectively, we give equal weight to both the components of the loss, i.e., we keep $\lambda = 1$ in eq. (4).

4.2 UAPs and the success rates

Similar to [13,17,18,15] we measure the effectiveness of the crafted UAPs in terms of their "success rate". It is the percentage of data samples (x) for which the target CNN predicts a different label upon adding the UAP (v). Note that

Table 1. Success rates of the perturbations modelled by our generative network, compared against the data-free approach FFF [17]. Rows indicate the target net for which perturbations are modelled and columns indicate the net under attack. Note that, in each row, entry where the target CNN matches with the network under attack represents white-box attack and the rest represent the black-box attacks. The mean fooling rate achieved by the Generator (G) trained for each of the target CNNs is shown in the rightmost column.

		VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152	Mean FR
VGG-F	Ours	**92.37**	**70.12**	**58.51**	**47.01**	**52.19**	**43.22**	**60.56**
	FFF	81.59	48.20	38.56	39.31	39.19	29.67	46.08
CaffeNet	Ours	**74.68**	**89.04**	**52.74**	**50.39**	**53.87**	**44.63**	**60.89**
	FFF	56.18	80.92	39.38	37.22	37.62	26.45	46.29
GoogLeNet	Ours	**57.90**	**62.72**	**75.28**	**59.12**	**48.61**	**47.81**	**58.57**
	FFF	49.73	46.84	56.44	40.91	40.17	25.31	43.23
VGG-16	Ours	**58.27**	**56.31**	**60.74**	**71.59**	**65.64**	**45.33**	**59.64**
	FFF	46.49	43.31	34.33	47.10	41.98	27.82	40.17
VGG-19	Ours	**62.49**	**59.62**	**68.79**	**69.45**	**72.84**	**51.74**	**64.15**
	FFF	39.91	37.95	30.71	38.19	43.62	26.34	36.12
ResNet-152	Ours	**52.11**	**57.16**	**56.41**	**47.21**	**48.78**	**60.72**	**53.73**
	FFF	28.31	29.67	23.48	19.23	17.15	29.78	24.60

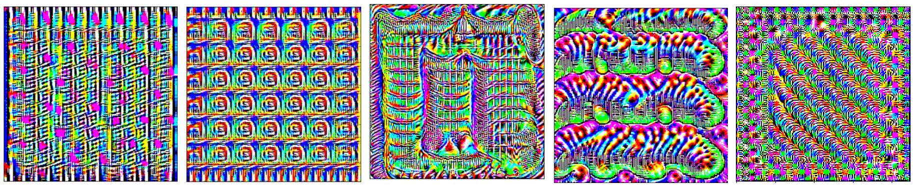


Fig. 4. Sample universal adversarial perturbations (UAP), learned by the proposed framework for different networks, the corresponding target CNN is mentioned below the UAP. Note that images shown are one sample for each of the target networks, and across different samplings the perturbations vary visually as shown in Fig. 6.

we compute the success rates over the 50000 validation images from ILSVRC dataset. Table 1 reports the obtained success rates of the UAPs crafted by our generative model G on various networks. Each row denotes the target model for which we train G and the columns indicate the model we attack to fool. Thus, we report the transfer rates on the unseen models also, which is referred to as "black-box attacking" (off-diagonal entries). Similarly, when the target CNN over which we learn G matches with the model under attack, it is referred to as "white-box attacking" (diagonal entries). Note that the right most column shows the mean success rates achieved by the individual generator networks (G) obtained across all the 6 CNN models. Proposed method can craft UAPs that have on an average 20.18% higher mean success rate compared to the existing data-free method to craft UAPs (FFF [17]).

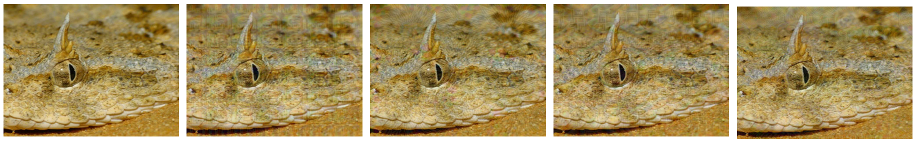
Figure 4 shows example UAPs learned by our approach for different target CNN models. Note that the pixel values in those perturbations lie in $[-10, 10]$. Also the UAPs for different models look different. Figure 5 shows a clean and corresponding perturbed samples after adding UAPs learned for different target CNNs. Note that each of the target CNNs misclassify them differently.

For the sake of completeness, we compare our approach with the data-driven counterpart also. Table 2 presents the white-box success rates for both data-free and data-driven methods to craft UAPs. We also show the fooling ability of random noise sampled in $[-10, 10]$ as a baseline. Note that the success rates obtained by random noise is very less compared to the learned UAPs. Thus the adversarial perturbations are highly structured and very effective compared to the performance of random noise as perturbation.

On the other hand, the proposed method of acquiring class impressions from the target model's memory increases the mean success rate by an absolute 20% from that of current state-of-the-art data-free approach (FFF [17]). Also, note that our approach performs close to the data-driven approach UAP [13] with a gap of 8%. These observations suggest that the class impressions are effective to serve the purpose of the actual data samples in the context of learning to craft the UAPs.

Table 2. Effectiveness of the proposed approach to handle the data absence. We compare the success rates against the data-driven approach UAP [13], data-free approach FFF [17] and random noise baseline.

	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152	Mean
Baseline	12.62	12.9	10.29	8.62	8.40	8.99	10.30
FFF (w/o Data)	81.59	80.92	56.44	47.10	43.62	29.78	56.58
Ours(w/o Data)	92.37	89.04	75.28	71.59	69.45	60.72	76.41
UAP (w Data)	93.8	93.1	78.5	77.8	80.8	84.0	84.67



Clean: Sand Viper — VGG-F: Maypole — CaffeNet: Afghan Hound — VGG19: Egyptian Cat — ResNet152: Chiton

Fig. 5. Clean image (leftmost) of class "Sand Viper", followed by adversarial images generated by adding UAPs crafted for various target CNNs. Note that the perturbations while remaining imperceptible are leading to different misclassifications.

4.3 Comparison with data dependent approaches.

Table 3 presents the transfer rates achieved by the image-agnostic perturbations crafted by the proposed approach. Each row denotes the target model on which the generative model (G) is learned and columns denotes the models under attack. Hence, diagonal entries denote the white-box adversarial attacks and the off diagonal entries denote the black-box attacks. Note that the main draft presents only the white-box success rates, for completeness we present both here. Also note that, in spite of being a data-free approach the mean SR (extreme right column) obtained by our method is very close to that achieved by the state-of-the-art data-driven approach to craft UAPs.

Table 3. Success rates (SR) for the perturbations crafted by the proposed approach compared against the state-of-the-art data driven approach for crafting the UAPs.

		VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152	Mean SR
VGG-F	Ours	92.37	70.12	58.51	47.01	52.19	43.22	60.56
	UAP	93.7	71.8	48.4	42.1	42.1	47.4	57.58
CaffeNet	Ours	74.68	89.04	52.74	50.39	53.87	44.63	60.89
	UAP	74.0	93.3	47.7	39.9	39.9	48.0	56.71
GoogLeNet	Ours	57.90	62.72	75.28	59.12	48.61	47.81	58.57
	UAP	46.2	43.8	78.9	39.2	39.8	45.5	48.9
VGG-16	Ours	58.27	56.31	60.74	71.59	65.64	45.33	59.64
	UAP	63.4	55.8	56.5	78.3	73.1	63.4	65.08
VGG-19	Ours	62.49	59.62	68.79	69.45	72.84	51.74	64.15
	UAP	64.0	57.2	53.6	73.5	77.8	58.0	64.01
ResNet-152	Ours	52.11	57.16	56.41	47.21	48.78	60.72	53.73
	UAP	46.3	46.3	50.5	47.0	45.5	84.0	53.27

4.4 Diversity

The objective of having the diversity component (L_d) in the loss is to avoid learning a single UAP and to learn a generative model that can generate diverse set of UAPs for a given target CNN. We examine the distribution of predicted labels after adding the generated UAPs. This can reveal if there is a set of sink labels that attract most of the predictions. We have considered the G learned to fool VGG-F model and 50000 samples of ILSVRC validation set. We randomly select 10 UAPs generated by the G and compute the mean histogram of predicted labels. After sorting the histogram, most of the predicted labels (95%) for proposed approach spread over 212 labels out of the total 1000 target labels. Whereas the same number for UAP [13] is 173. The observed 22.5% higher diversity is attributed to our diversity component (L_d).

4.5 Simultaneous Targets

The ability of the adversarial perturbations to generalize across multiple models is observed with both image-specific ([28,7]) and agnostic perturbations ([13,17]). It is an important issue to be investigated since it makes simple black-box attacks possible via transferring the perturbations to unknown models. In this subsection we investigate to learn a single G that can can craft UAPs to simultaneously fool multiple target CNNs.

Table 4. Generalizability of the UAPs crafted by the ensemble generator G_E learned on three target CNNs: CaffeNet, VGG-16 and ResNet-152. Note that because of the ensemble of the target CNNs, G_E learns to craft perturbations that have higher mean black-box success rates (MBBSR) compared to that of the individual generators.

	G_C	G_{V16}	G_{R152}	G_E
MBBSR	60.34	61.46	52.43	**68.52**

We replace the single target CNN with an ensemble of three models: CaffeNet, VGG-16 and ResNet-152 and learn G_E using the fooling and diversity losses. Note that, since the class impressions vary from model to model, for this experiment we generate class impressions from multiple CNNs. Particularly, we simultaneously maximize the pre-softmax activation (eq.(1)) of the desired class across individual target CNNs via optimizing their mean. We then investigate the generalizability of the generated perturbations. Table 4 presents the mean black-box success rate (MBBSR) for the UAPs generated by G_E on the remaining 3 models. For comparison, we present the MBBSR of the generators learned on the individual models. Because of the ensemble of the target CNNs G_E learns to craft more general UAPs and therefore achieves higher success rates than the individual generators.

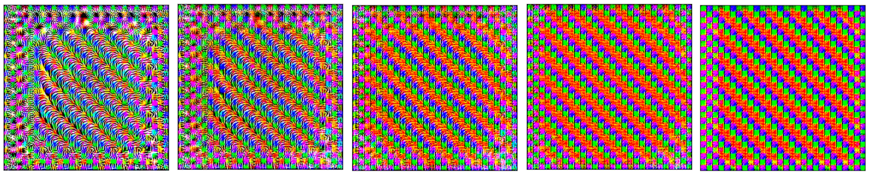
4.6 Interpolating in the latent space

Our generator network (G) is similar to that in a typical GAN [6,22]. It maps the latent space to the space of UAPs for the given target classifier(s). In case of GANs, interpolating in the latent space can reveal signs of memorization. While traversing the latent space, smooth semantic change in the generations means the model has learned relevant representations. In our case, since we generate UAPs, we investigate if the interpolation has smooth visual changes and the intermediate UAPs can also fool the target CNN coherently.

Figure 6 shows the results of interpolating in the latent space for ResNet-152 as the target CNN. We sample a pair of points (z_1 and z_2) in the latent space and consider 5 intermediate points on the line joining them. We generate the UAPs corresponding to all these points by passing them through the learned generator architecture G. Figure 6 shows the generated UAPs and the corresponding success rates in fooling the target CNN. Note that the UAPs change visually smoothly between any pair of points and the success rate remains unchanged. This ensures that the representations learned are relevant and interesting.

4.7 Adversarial Training

We have performed adversarial training of target CNN with 50% mixture of clean and adversarial samples crafted using the learned generator (G). After 2 epochs, success rate of the G has dropped from 75.28 to 62.51. Note that the improvement is minor and the target CNN is still vulnerable. We then repeated



$0.0*z_1+1.0*z_2$: 60.58

$0.25*z_1+0.75*z_2$: 59.16

$0.5*z_1+0.5*z_2$: 60.25

$0.75*z_1+0.25*z_2$: 59.87

$1.0*z_1+0.0*z_2$: 60.09

Fig. 6. Interpolation between a pair of points in Z space shows that the mapping learned by our generator has smooth transitions. The figure shows the perturbations corresponding to 5 points on the line joining a pair of points (z_1 and z_2) in the latent space. Note that these perturbations are learned to fool the ResNet-152 [8] architecture. Below each perturbation, the corresponding success rate obtained over 50000 images from ILSVRC 2014 validation images is mentioned. This shows the fooling capability of these intermediate perturbations is also high and remains same at different locations.

the generator training for the finetuned network, resulting generator fools the finetuned network with an increased success rate of 68.72. After repeating this for multiple iterations, we observe that adversarial training does not make the target CNN significantly robust.

5 Discussion and Conclusions

In this paper we have presented a novel approach to mitigate the absence of data for crafting Universal Adversarial Perturbations (UAP). Class impressions are representative images that are easy to obtain via simple optimization from the target model. Using class impressions, our method drastically reduces the performance gap between the data-driven and data-free approaches to craft the UAPs. Success rates closer to that of data-driven UAPs demonstrate the effectiveness of class impressions in the context of crafting UAPs.

Another way to look at this observation is that it would be possible to extract useful information about the training data from the model parameters in a task specific manner. In this paper, we have extracted the class impressions as proxy data samples to train a generative model that can craft UAPs for the given target CNN classifier. It would be interesting to explore such feasibility for other applications as well. Particularly, we would like to investigate if the existing adversarial setup of the GANs might get benefited with any additional information extracted from the discriminator network and generate more natural looking synthetic data.

The generative model presented in our approach is an efficient way to craft UAPs. Unlike the existing methods that perform complex optimizations, our approach constructs UAPs through a simple feed forward operation. Significant success rates, surprising cross model generalizability even in the absence of data reveal severe susceptibilities of the current deep learning models.

References

1. Abadi et al., M.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), http://tensorflow.org/, software available from tensorflow.org 7, 9
2. Baluja, S., Fischer, I.: Learning to attack: Adversarial transformation networks. In: Proceedings of AAAI (2018) 4
3. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 387–402 (2013) 1
4. Biggio, B., Fumera, G., Roli, F.: Pattern recognition systems under attack: Design issues and research challenges. International Journal of Pattern Recognition and Artificial Intelligence **28**(07) (2014) 1
5. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. In: Proceedings of the British Machine Vision Conference (BMVC) (2014) 6, 7
6. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, (NIPS) (2014) 8, 13
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR) (2015) 1, 4, 8, 12
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015) 14
9. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D.: Adversarial machine learning. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence. AISec '11 (2011) 1
10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 7
11. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: International Conference on Learning Representations (ICLR) (2017) 1
12. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: International Conference on Learning Representations (ICLR) (2017) 2
13. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2, 3, 4, 5, 9, 11, 12
14. Moosavi-Dezfooli, S., Fawzi, A., Frossard, P.: Deepfool: A simple and accurate method to fool deep neural networks. In: IEEE conference on Computer Vision and Pattern Recognition (CVPR) (2016) 1, 2, 4
15. Mopuri, K.R., Ganeshan, A., Babu, R.V.: Generalizable data-free objective for crafting universal adversarial perturbations. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018) 2, 3, 4, 9
16. Mopuri, K.R., Garg, U., Babu, R.V.: CNN fixations: An unraveling approach to visualize the discriminative image regions. arXiv preprint arXiv:1708.06670 (2017) 4
17. Mopuri, K.R., Garg, U., Babu, R.V.: Fast feature fool: A data independent approach to universal adversarial perturbations. In: Proceedings of the British Machine Vision Conference (BMVC) (2017) 2, 3, 4, 5, 9, 10, 11, 12

18. Mopuri, K.R., Ojha, U., Garg, U., Babu, R.V.: NAG: Network for adversary generation. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) (2018) 4, 5, 8, 9
19. Mordvintsev, A., Tyka, M., Olah, C.: Google deep dream (2015), https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html 4
20. Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill (2017), https://distill.pub/2017/feature-visualization 4, 5
21. Papernot, N., McDaniel, P.D., Goodfellow, I.J., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. In: Asia Conference on Computer and Communications Security (ASIACCS) (2017) 2, 8
22. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015) 13
23. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015) 9
24. Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems (NIPS) (2016) 9
25. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: The IEEE International Conference on Computer Vision (ICCV) (2017) 4
26. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: International Conference on Learning Representations ICLR Workshops (2014) 4, 5
27. Springenberg, J., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: International Conference on Learning Representations (ICLR) (workshop track) (2015) 4
28. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR) (2013) 1, 2, 4, 12
29. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (ECCV). pp. 818–833 (2014) 4
30. Zhang, J., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. In: European Conference on Computer Vision(ECCV) (2016) 4
31. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of Computer Vision and Pattern Recognition (CVPR) (2016) 4