NAG: Network for Adversary Generation

Konda Reddy Mopuri*, Utkarsh Ojha*, Utsav Garg and R. Venkatesh Babu
Video Analytics Lab, Computational and Data Sciences
Indian Institute of Science, Bangalore, India

Abstract

Adversarial perturbations can pose a serious threat for deploying machine learning systems. Recent works have shown existence of image-agnostic perturbations that can fool classifiers over most natural images. Existing methods present optimization approaches that solve for a fooling objective with an imperceptibility constraint to craft the perturbations. However, for a given classifier, they generate one perturbation at a time, which is a single instance from the manifold of adversarial perturbations. Also, in order to build robust models, it is essential to explore the manifold of adversarial perturbations. In this paper, we propose for the first time, a generative approach to model the distribution of adversarial perturbations. The architecture of the proposed model is inspired from that of GANs and is trained using fooling and diversity objectives. Our trained generator network attempts to capture the distribution of adversarial perturbations for a given classifier and readily generates a wide variety of such perturbations. Our experimental evaluation demonstrates that perturbations crafted by our model (i) achieve state-of-the-art fooling rates, (ii) exhibit wide variety and (iii) deliver excellent cross model generalizability. Our work can be deemed as an important step in the process of inferring about the complex manifolds of adversarial perturbations.

1. Introduction

Machine learning systems are shown [5, 4, 11] to be vulnerable to adversarial noise: small but structured perturbation added to the input that affects the model's prediction drastically. Recently, the most successful Deep Neural Network based object classifiers have also been observed [29, 9, 19, 16, 22] to be susceptible to adversarial attacks with almost imperceptible perturbations. Researchers have attempted to explain this intriguing aspect via hypothesizing linear behaviour of the models (*e.g.* [9, 19]), finite training data (*e.g.* [3]), etc. More importantly, the adversar-

---

* Equal contribution.

ial perturbations exhibit cross model generalizability. That is, the perturbations learned on one model can fool another model even if the second model has a different architecture or has been trained with different dataset [29, 9].

Recent startling findings by Moosavi-Dezfooli *et al.* [18] and Mopuri *et al.* [21, 20] have shown that it is possible to mislead multiple state-of-the-art deep neural networks over most of the images by adding a single perturbation. That is, these perturbations are image-agnostic and can fool multiple diverse networks trained on a target dataset. Such perturbations are named "Universal Adversarial Perturbations" (UAP), because a single adversarial noise can perturb images from multiple classes. On one side, the adversarial noise poses a severe threat for deploying machine learning based systems in the real world. Particularly, for the applications that involve safety and privacy (*e.g.*, autonomous driving and access granting), it is essential to develop robust models against adversarial attacks. On the other side, it also poses a challenge to our understanding of these models and the current learning practices. Thus, the adversarial behaviour of the deep learning models to small and structured noise demands a rigorous study now more than ever.

All the existing methods, weather image specific [29, 9, 19, 6, 17] or agnostic [18, 21, 20], can craft only a single perturbation that makes the target classifier susceptible. Specifically, these methods typically learn a single perturbation (δ) from a possibly bigger set of perturbations (Δ) that can fool the target classifier. It is observed that for a given technique (*e.g.*, UAP [18], FFF [21]), the perturbations learned across multiple runs are not very different. In spite of optimizing with a different data ordering or initialization, their objectives end up learning very close perturbations in the space (refer sec. 3.3). In essence, these approaches can only prove that the UAPs exist for a given classifier by crafting one such perturbation (δ). This is very limited information about the underlying distribution of such perturbations and in turn about the target classifier itself. Therefore, a more relevant task at hand is to model the distribution of adversarial perturbations. Doing so can help us better analyze the susceptibility of the models against ad-
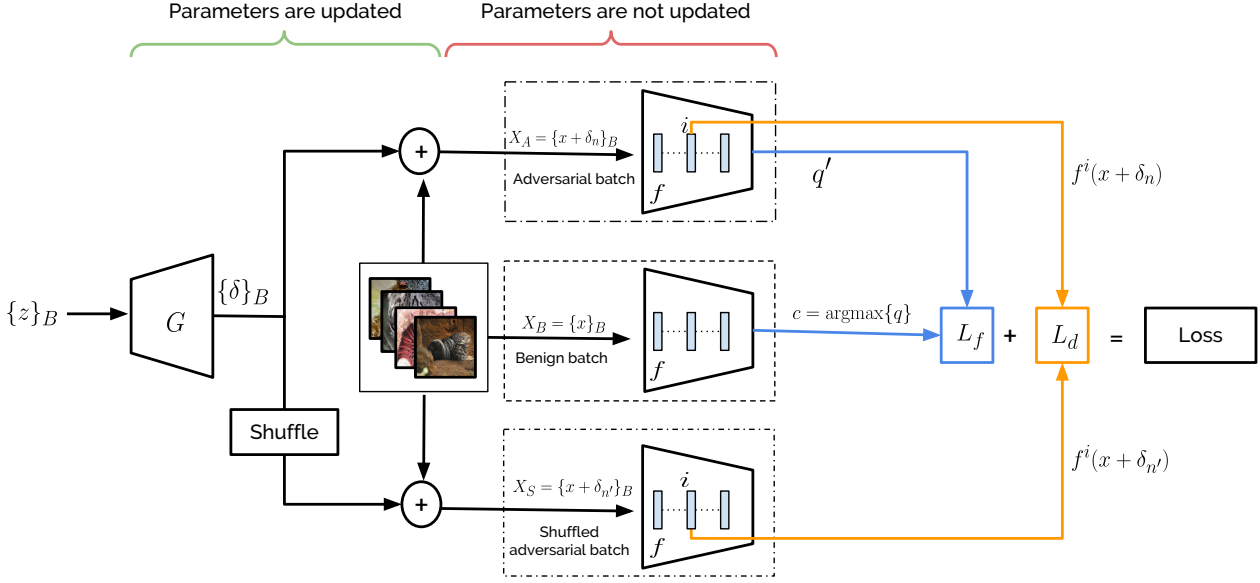
Figure 1. Overview of the proposed approach that models the distribution of universal adversarial perturbations for a given classifier. The illustration shows a batch of B random vectors $\{z\}_B$ transforming into perturbations $\{\delta\}_B$ by G which get added to the batch of data samples $\{x\}_B$. The top portion shows *adversarial batch* (X_A), bottom portion shows *shuffled adversarial batch* (X_S) and middle portion shows the *benign batch* (X_B). The *Fooling objective* L_f (eq. 2) and *Diversity objective* L_d (eq. 3) constitute the loss. Note that the target CNN (f) is a trained classifier and its parameters are not updated during the proposed training. On the other hand, the parameters of generator (G) are randomly initialized and learned through backpropagating the loss. (Best viewed in color).

versarial perturbations. Furthermore, modelling the distributions would provide insights regarding the transferability of adversarial examples and help to prevent black-box attacks [23, 17]. It also helps to efficiently generate large number of adversarial examples for learning robust models via adversarial training [30].

Empirical evidence [9, 31] has shown that the perturbations exist in large contiguous regions rather than being scattered in multiple small discontinuous pockets. In this paper, we attempt to model such regions for a given classifier via generative modelling. We introduce a GAN [8] like generative model to capture the distribution of the unknown adversarial perturbations. The freedom from parametric assumptions on the distribution and the target distribution being unknown (no known samples from the target distribution of adversarial perturbations) make the GAN framework a suitable choice for our task.

The major contributions of this work are:

- A novel objective (eq. 2) to craft universal adversarial perturbations for a given classifier that achieves the state-of-the art fooling performance on multiple CNN architectures trained for object recognition.

- For the first time, we show that it is possible to model the distribution (Δ) of such perturbations for a given

classifier via a generative model. For this, we present an easily trainable framework for modelling the unknown distribution of perturbations.

- We demonstrate empirically that the learned model can capture the distribution of perturbations and generates perturbations that exhibit diversity, high fooling capacity and excellent cross model generalizability.

The rest of the paper is organized as follows: section 2 details the proposed method, section 3 presents comprehensive experimentation to validate the utility of the proposed method, section 4 briefly discusses the existing related works, and finally section 5 concludes the paper.

2. Proposed Approach

This section presents a detailed account of the proposed method. For ease of reference, we first briefly introduce the GAN [8] framework.

2.1. GANs

Generative models for images have seen renaissance lately, especially because of the availability of large datasets [25, 32] and the emergence of deep neural networks. Particularly, Generative Adversarial Networks

(GAN) [8] and Variational Auto-Encoders (VAE) [15] have shown significant promise in this direction. In this work, we utilize a GAN like framework to model the distribution of the adversarial perturbations.

A typical GAN framework consists of two parts: a Generator (G) and a Discriminator (D). The generator G transforms a random vector z into a meaningful image I; i.e., $G(z) = I$, where z is usually sampled from a simple distribution (e.g., $\mathcal{N}(0,1), \mathcal{U}(-1,1)$). G is trained to produce images (I) that are indistinguishable from real images from the true data distribution p_{data}. The discriminator D accepts an image and outputs the probability for it to be a real image, a sample from p_{data}. Typically, D is trained to output low probability p_D when a fake (generated) image is presented. Both G and D are trained adversarially to compete with and improve each other. A properly trained generator G at the end of training is expected to produce images that are indistinguishable from real images.

2.2. Modelling the adversaries

A broad overview of our method is illustrated in Fig. 1. We first formalize the notations used in the subsequent sections of the paper. Note that in this paper, we have considered CNNs that are trained for object recognition [10, 28, 27, 13]. The data distribution over which the classifiers are trained is denoted as $\mathcal{X}$ and a particular sample from $\mathcal{X}$ is represented as x. The target CNN is denoted as f, therefore the output of a given layer i is denoted as $f^i(x)$. The predicted label for a given data sample x is denoted as $\hat{k}(x)$. Output of the softmax layer is denoted as q, which is a vector of predicted probabilities q_j for each of the target categories j. The image-agnostic additive perturbation that fools the target CNN is denoted as δ. ξ denotes the limit on the perturbation (δ) in terms of its l_∞ norm. Our objective is to model the distribution of such perturbations (Δ) for a given classifier. Formally, we seek to model

$$\Delta = \{\delta : \hat{k}(x + \delta) \neq \hat{k}(x) \text{ for } x \sim \mathcal{X} \text{ and} \\ || \delta ||_\infty < \xi\} \quad (1)$$

Since our objective is to model the unknown distribution of image-agnostic perturbations for a given trained classifier (target CNN), we make suitable changes in the GAN framework. The modifications we make are: (i) Discriminator (D) is replaced by the target CNN (f) which is already trained and whose weights are frozen, and (ii) a novel loss (fooling and diversity objectives) instead of the adversarial loss to train the Generator (G). Thus, the objective of our work is to train a model (G) that can fool the target CNN. The architecture for G is also similar to that of typical GAN which transforms a random sample to an image through a dense layer and a series of deconv layers. More details about the exact architecture are discussed in section 3. We now proceed to discuss the fooling objective

that enables us to model the adversarial perturbations for a given classifier.

2.3. Fooling Objective

In order to fool the target CNN, the generator G should be driven by a suitable objective. Typical GANs use adversarial loss to train their G. However, in this work we attempt to model a distribution whose samples are unavailable. We know only a single attribute of those samples which is *to be able to fool the target classifier*. We incorporate this attribute via a fooling objective to train our G that models the unknown distribution (Δ) of perturbations.

We denote the label predicted by the target CNN on a clean sample x as *benign prediction* (c) and that predicted on the corresponding perturbed sample ($x + \delta$) as *adversarial prediction*. Similarly, we denote the output vector of the softmax layer without δ and after adding δ as q and q' respectively. Ideally a perturbation δ should confuse the classifier so as to flip the benign prediction into a different adversarial prediction. For this to happen, after adding δ, the confidence of the benign prediction (q'_c) should be reduced and that of another category should be made higher. Thus, we formulate a fooling loss to minimize the confidence of benign prediction on the perturbed sample ($x + \delta$)

$$L_f = -log(1 - q'_c) \quad (2)$$

Fig. 1 shows a graphical explanation of the objective, where the fooling objective is shown by the blue colored block. Note that the fooling loss essentially encourages G to generate perturbations that decrease confidence of benign predictions and thus eventually flip the label.

2.4. Diversity Objective

The fooling loss only encourages to learn a G that can guarantee high fooling capability for the generated perturbations (δ). This objective might lead to some local minima where the G learns only a limited set of effective perturbations as in [18, 21]. However, our objective is to model the distribution Δ such that it covers all varieties of those perturbations. Therefore, we introduce an additional component to the loss that encourages G to explore the space of perturbations and generate a diverse set of perturbations. We term this objective the *Diversity objective*. Within a mini-batch of generated perturbations, this objective indirectly encourages them to be different by separating their feature embeddings projected by the target classifier. In other words, for a given pair of generated perturbations δ_n and δ'_n, our objective increases the distance between $f^i(x + \delta_n)$ and $f^i(x + \delta'_n)$ at a given layer i in the classifier.

$$L_d = -\sum_{n=1}^{B} d(f^i(x_n + \delta_n), f^i(x_n + \delta_{n'})) \quad (3)$$

where n' is a random index in $[1, B]$ and $n' \neq n$, B is the batch size, x_n, δ_n are n^{th} data sample and perturbation in the mini-batch respectively. Note that a batch contains B perturbations (δ) generated by G (via transforming random vectors (z)) and B data samples (x). f^i is the output of the CNN at i^{th} layer and $d(.,.)$ is a distance metric (*e.g.*, Euclidean) between a pair of features. The orange colored block in Fig. 1 illustrates the diversity objective.

Thus, our final loss becomes the summation of both fooling and diversity objectives and is given by

$$Loss = L_f + \lambda L_d \qquad (4)$$

Since it is important to learn diverse perturbations that exhibit maximum fooling, we give equal importance to both L_f and L_d in the final loss to learn the G (i.e., $\lambda = 1$).

2.5. Architecture and Implementation details

Before we present the experimental details, we describe the implementation and working details of the proposed architecture. The generator part (G) of the network maps the latent space Z to the distribution of perturbations (Δ) for a given target classifier. The architecture of the generator consists of 5 deconv layers. The final deconv layer is followed by a $tanh$ non-linearity and scaling by ξ. Doing so restricts the perturbations' range to $[-\xi, \xi]$. Following [18, 21, 20], the value of ξ is chosen to be 10 in order to add a quasi-imperceptible adversarial noise. The generator network is adapted from [26]. We performed all our experiments on a variety of CNN architectures trained to perform object recognition task on the ILSVRC-2014 [25] dataset. We kept the architecture of our generator (G) unchanged for different target CNN architectures and separately learned the corresponding adversarial distributions.

During training, we sample a batch of random vectors $z \in \mathcal{R}^d$ from the uniform distribution $\mathcal{U}[-1, 1]$ which in turn get transformed by G into a batch of perturbations $\{\delta\}_B = \{\delta_1, \delta_2, \delta_3, \ldots, \delta_B\}$ each of size equal to that of the image (*e.g.*, $224 \times 224 \times 3$). We also sample B images $\{x\}_B = \{x_1, x_2, x_3, \ldots, x_B\}$ from the available training data to form the mini-batch training data, denoted as *benign batch* (X_B). We now add the perturbations to the training data in a one-to-one manner i.e. one perturbation gets added to the corresponding image of the batch, which gives us the *adversarial batch*, $X_A = \{x_1 + \delta_1, x_2 + \delta_2, x_3 + \delta_3, \ldots, x_B + \delta_B\}$. This is shown in the top portion of Fig. 1. We also randomly shuffle the perturbations ensuring no perturbation remains in its original index in the batch, i.e., $\{\delta_1', \delta_2', \delta_3', \ldots, \delta_B'\}$ such that $\delta_i \neq \delta_i', \forall i$. With this, we form a *shuffled adversarial batch* as $X_S = \{x_1 + \delta_1', x_2 + \delta_2', x_3 + \delta_3', \ldots, x_B + \delta_B'\}$, which is shown in the bottom portion of Fig. 1. Note that in order to prepare X_S, only the perturbations ($\{\delta\}_B$) are shuffled but not the data samples ($\{x\}_B$).

Thus, each of our training iterations consists of three quasi-batches, namely, (i) Benign images batch X_B, (ii) Adversarial batch X_A, and (iii) Shuffled adversarial batch X_S. These are the three portions shown in Fig. 1. We now feed these through the target CNN (f) and compute the loss. We obtain the benign predictions (c) over the clean batch samples $\{x\}_B$. These labels are used to compute the confidences (q_c') for the corresponding adversarial batch samples. This forms the fooling objective as shown in eq. 2. Similarly, we obtain the feature representations at the softmax layer (probability vectors) for both adversarial and shuffled adversarial batches (top and bottom portions of Fig. 1) to compute the diversity component of the loss as shown in eq. 3. Essentially, our diversity objective pushes apart the final layer representations corresponding to two different perturbations (δ_i and δ_i') via maximizing the cosine distance between them.

Note that we update only the G part of the network and the target CNN, which is a pretrained classifier under attack, remains unchanged. We iteratively perform the loss computation and parameter updation for all the samples in the training data. During training, we use a small held-out set of 1000 random images as validation set and stop our training upon reaching best performance on this set. In our experiments, the maximum number of epochs is set to 100 and we observe that training of generators for all the target CNNs gets saturated at around $60 - 70$ epochs.

3. Experiments

For all our experiments, we worked with 10000 (10 per category, similar to [18]) training images randomly chosen from ILSVRC 2014 train set and 50000 images of ILSVRC 2014 validation set as our testing images. The latent space dimension d is set to 10. We have experimented with spaces of different dimensions (*e.g.*, 50, 100) and observed that the fooling rates obtained are very close. However, we observe the generated perturbations for $d = 10$ demonstrate larger visual variety than other cases. Thus, we keep $d = 10$ for all our experiments. We use a batch size (B) of 64 for shallow networks such as VGG-F [7] and GoogLeNet [28], and 32 for the rest. The models are implemented in TensorFlow [1] with Adam optimizer [14] on a TITAN-X GPU card. Codes for the project are available at https://github.com/val-iisc/nag.

3.1. Perturbations and the fooling rates

The fooling rates achieved by the perturbations crafted by the learned generative model (G) are presented in Table 1. Results are shown for seven different network architectures trained on ILSVRC-2014 dataset computed for over 50000 test images. We also investigate the transfer rates of the perturbations by attacking other unknown models along with the target CNN. Rows denote a particular

Table 1. Average fooling rates of the perturbations modelled by our generative network vs. UAP [18]. 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. For our method, along with average fooling rates, the corresponding standard deviations are also mentioned. The best result for each case is shown in bold and UAP best cases are shown in *blue*. Mean avg. fooling rate achieved by the Generator (G) for each of the target CNNs is shown in the rightmost column.

		VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-50	ResNet-152	Mean FR
VGG-F	Our	**94.10 ± 1.84**	**81.28**± 3.50	**64.15**±3.41	**52.93**±8.50	**55.39**±2.68	50.56±4.50	**47.67**±4.12	**63.73**
	UAP	93.7	71.8	48.4	42.1	42.1	-	47.4	57.58
CaffeNet	Our	**79.25**±1.44	**96.44**±1.56	**66.66**±1.84	**50.40**±5.61	**55.13**±4.15	52.38±3.96	**48.58**±4.25	**64.12**
	UAP	74.0	93.3	47.7	39.9	39.9	-	48.0	56.71
GoogLeNet	Our	**64.83**±0.86	**70.46**±2.12	**90.37**±1.55	**56.40**±4.13	**59.14**±3.17	63.21±4.40	**59.22**±1.64	**66.23**
	UAP	46.2	43.8	78.9	39.2	39.8	-	45.5	48.9
VGG-16	Our	60.56±2.24	**65.55**±6.95	**67.38**±4.84	77.57±2.77	**73.25**±1.63	61.28±3.47	54.38±2.63	**65.71**
	UAP	*63.4*	55.8	56.5	*78.3*	73.1	-	*63.4*	65.08
VGG-19	Our	**67.80**±2.49	**67.58**±5.59	**74.48**±0.94	**80.56**±3.26	**83.78**±2.45	68.75±3.38	**65.43**±1.90	**72.62**
	UAP	64.0	57.2	53.6	73.5	77.8	-	58.0	64.01
ResNet-50	Our	47.06±2.60	63.35±1.70	65.30±1.14	55.16±2.61	52.67±2.58	86.64±2.73	66.40±1.89	62.37
	UAP	-	-	-	-	-	-	-	-
ResNet-152	Our	**57.66**±4.37	**64.86**±2.95	**62.33**±1.39	**52.17**±3.41	**53.18**±4.16	73.32±2.75	**87.24**±2.72	**64.39**
	UAP	46.3	46.3	50.5	47.0	45.5	-	84.0	53.27

target CNN for which we have modelled the distribution of perturbations and the columns represent the classifiers we attack. Note that in each row, when the target CNN (row) matches with the system under attack (column), the fooling indicates the white-box attack scenario and all other entries represent the black-box attack scenario.

Since our G network models the perturbation space, we can now easily generate a perturbation by sampling a z and feeding it through the G. In Table 1, we report the mean fooling rates after generating multiple perturbations for a given CNN classifier. Particularly, the white-box fooling rates are computed by averaging over 100 perturbations and black-box rates are averaged over 10. The standard deviations are mentioned next to the fooling rates. Also, the mean average fooling rate achieved by the learned model (G) for each of the target CNNs is shown in the rightmost column. Clearly, the proposed generative model captures the perturbations with higher fooling rates than UAP [18]. Note that of all the 36 entries for which UAP [18] provided their fooling rates, in only 3 cases (indicated in bold faced blue in Table 1) they perform better than us. The mean fooling rate (of all the entries in the table, except the rightmost column) obtained by the UAP [18] is 57.66 and that achieved by our model is 65.68, which is a significant 8% improvement.

Figure 2 shows the perturbations generated by the proposed generative model for different target CNNs. Note that each of them is one random sample from the corresponding distributions of perturbations. Fig. 4 shows a benign sample and the corresponding perturbed samples after adding perturbations for multiple CNNs. Note that the perturbations are sampled from the corresponding distributions learned by our method.

Table 2. Effect of training data on the modelling.

	1000	2000	4000	10000	50000
White-box	61.54	73.19	78.18	87.24	91.16
Mean BBFR	39.46	45.12	51.87	62.94	67.45

3.2. Effect of training data on the modelling

In this subsection, we examine the effect of available training data on the learning. We have considered ResNet-152 model and various training data sizes (equal population from each category). Table 2 presents the fooling rates obtained by the crafted perturbations in both white-box and black-box setup. Note that the black-box fooling rates are obtained by averaging the fooling rates obtained on three (GoogLeNet, VGG-19 and ResNet-50) CNN models. As one expects, owing to better modelling induced by availability of more training data, the fooling rates increase with available training data.

3.3. Diversity of perturbations

We examine the diversity of the generated perturbations by our model. It can be interesting to examine the predicted label distribution after adding the perturbations. Doing so can reveal if there exist any dominant labels that most of the images get confused to or whether the confusions are diverse. In this subsection, we analyze the labels predicted by the target CNN after adding perturbations modelled by the corresponding generative models (G). We have considered VGG-F architecture and the 50000 images from the validation set of ILSVRC-2014. We compute the mean histogram of the predicted labels for 10 perturbations generated by our G. The top-5 categories are: {jigsaw puzzle, maypole, otter, dome, electric fan}. Though there exists a slight domination from some of the categories, the extent of domination is far less compared to [18]. While in our case, 257

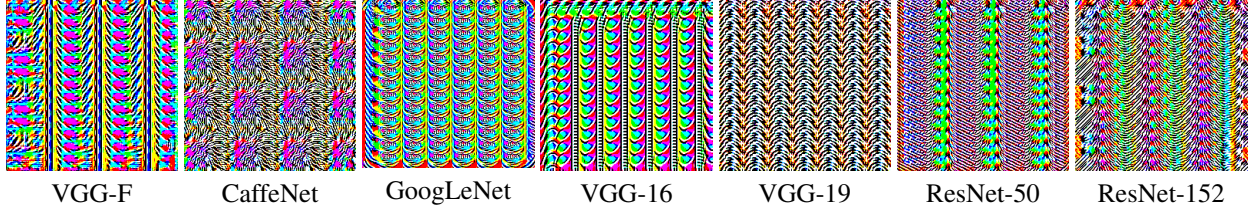VGG-F   CaffeNet   GoogLeNet   VGG-16   VGG-19   ResNet-50   ResNet-152

Figure 2. Sample universal adversarial perturbations for different networks. The target CNN is mentioned below the perturbations. Note that these are one sample from each of the corresponding distributions and across different samplings of the same generative model the perturbations vary visually.
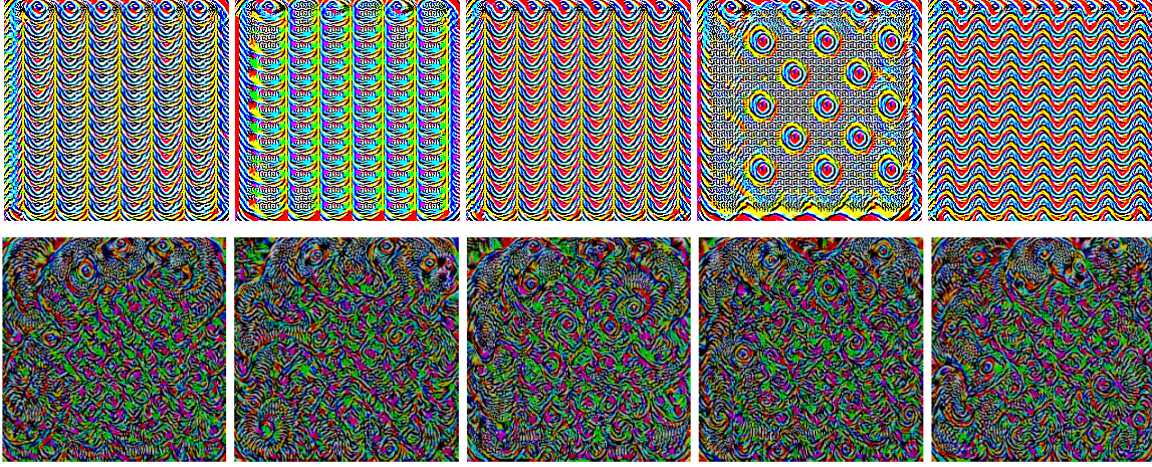

Figure 3. Sample perturbations generated by proposed approach (top) and UAP [18] (bottom) for GoogLeNet [28]. Note that the perturbations generated by [18] look very similar to each other, whereas generated by our approach showcase diversity. These results show that the proposed method faithfully models the distribution of perturbations that can effectively fool the target CNN.

categories account for the 95% of the predicted labels, for UAP [18], it is 173 categories. The 48.6% relative higher diversity compared to [18] is attributed to the effectiveness of the proposed diversity loss (eq. 3) which encourages the model to explore various regions in the adversarial manifold.

3.4. Traversing the manifold of perturbations

In this subsection, we perform experiments to understand the landscape of the latent space (Z). In case of GANs, traversing on the learned manifold generally tells about the signs of memorization [24]. While walking on the latent space, if the image generations result in semantic changes, it is considered that the model has learned relevant and interesting representations. However, our generative model attempts to learn the unknown distribution of adversarial perturbations with no samples from the target distribution. Therefore, it is not relevant to investigate for the smooth semantic changes in the generations but only to look for smooth visual changes, while retaining the ability to fool the target classifier.

Figure 5 shows the results of interpolation experiments on the ResNet-152 [10] classifier. We have randomly taken

a pair of points (z_1 and z_2) in the latent (Z) space and considered 10 intermediate points on the line joining z_1 and z_2. We have generated the perturbations corresponding to these intermediate points by feeding them through the learned generative model (G). Figure 5 shows the generated perturbations at the intermediate points along with their fooling rates. We clearly observe that the perturbations change smoothly between any pair of consecutive points and the sequence gives a morphing like effect with large number of intermediate points. For each of the intermediate perturbations, fooling rate is computed over the 50000 images from the ILSVRC-2014 validation set. In Fig. 5, below each of these perturbations, corresponding fooling rates are mentioned. The high and consistent fooling rates along the path demonstrate that the modelling of the adversarial distribution has been faithful. The proposed approach generates perturbations smoothly from the underlying manifold. We attribute this ability of our learned generative model to the effectiveness of the proposed objectives in the loss.

3.5. Modelling adversaries for multiple targets

Transferability of the adversarial perturbations (both image specific and agnostic) has been an intriguing revelation
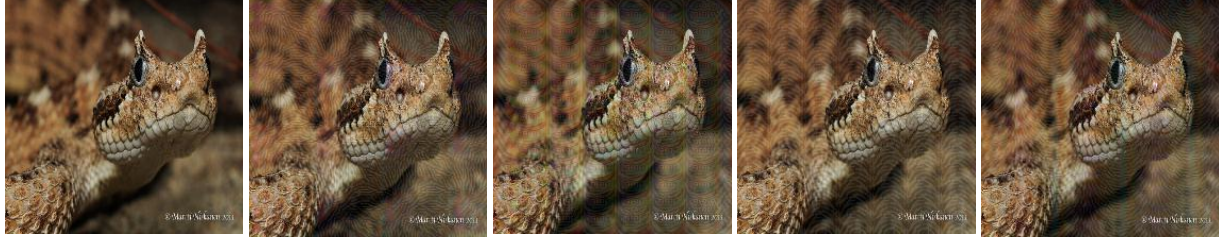
| Clean : Horned viper | CaffeNet : N. Terrier | GoogLeNet : Hamper | VGG-19 : Hyena | ResNet-152 : Chiton |

Figure 4. A clean image (left most) and corresponding adversarial images crafted for multiple networks along with predictions.

| $0.0 * z_1 + 1.0 * z_2$: 87.53 | $0.1 * z_1 + 0.9 * z_2$: 87.71 | $0.2 * z_1 + 0.8 * z_2$: 87.73 | $0.3 * z_1 + 0.7 * z_2$: 87.25 | $0.4 * z_1 + 0.6 * z_2$: 87.48 |

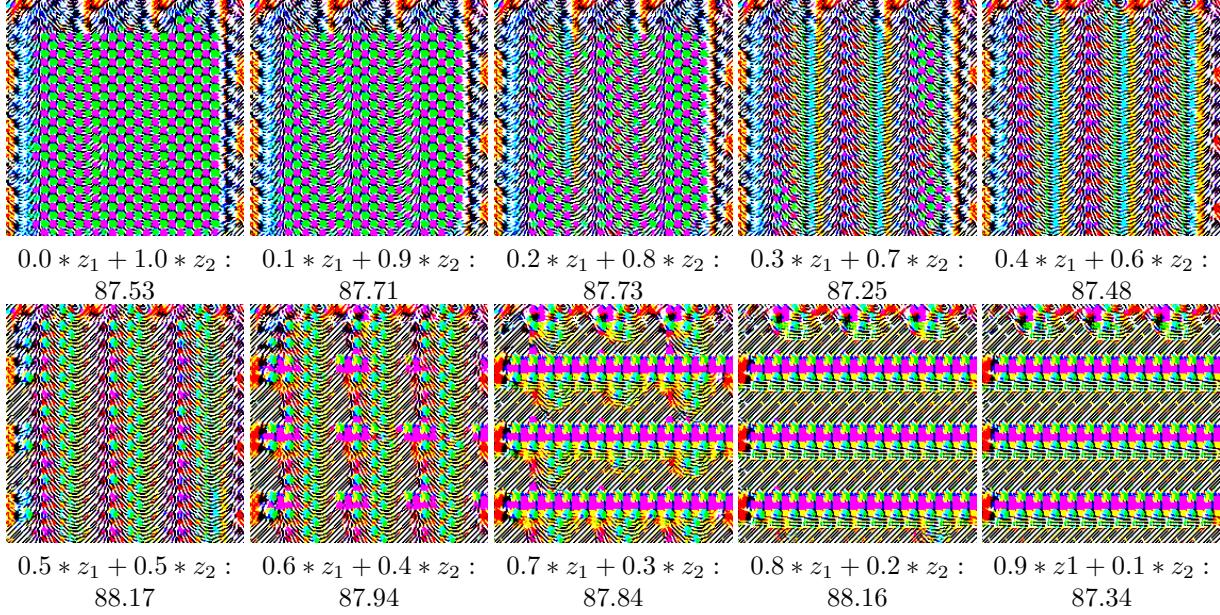| $0.5 * z_1 + 0.5 * z_2$: 88.17 | $0.6 * z_1 + 0.4 * z_2$: 87.94 | $0.7 * z_1 + 0.3 * z_2$: 87.84 | $0.8 * z_1 + 0.2 * z_2$: 88.16 | $0.9 * z1 + 0.1 * z_2$: 87.34 |

Figure 5. Interpolation between a pair of points in Z space shows that the distribution learned by our generator has smooth transitions. The figure shows the perturbations corresponding to 10 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 [10] architecture. Below each perturbation, the corresponding fooling 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 in the learned distribution of perturbations.

by many of the recent adversarial perturbations works [29, 9, 22, 18, 21, 20]. Moosavi-Dezfooli *et al.* [18] attempted to explain the cross-model generalizability with the correlation among different regions in the classification boundaries learned by them. In this subsection, we investigate if we can learn to model a single distribution of adversaries that fool multiple target CNNs simultaneously.

We consider all the 7 target CNNs presented in Table 1 to model a single adversarial manifold that can fool all of them simultaneously. We keep the G part of the proposed architecture unchanged while we replace single target classifier with all the target networks. Because of the memory constraint to fit all the models, we train with a smaller batch size. The loss to train the generator is the summation of the individual losses (eq. 4) computed for each of the target CNNs separately. Thus, the objective driving the optimization aims to craft the perturbations that fool all the target

CNNs simultaneously. Similar to the single target case, the diversity objective (eq. 3) encourages to explore multiple regions covering the manifold of perturbations and model a distribution with a lot of variety.

Table 3 presents the mean fooling rates obtained by 10 samples from the distribution of perturbations learned to fool all the 7 target CNNs. The fooling rates are slightly lesser than those obtained for the dedicated optimization (white-box attacks in Tab. 1). However, given the complexity of the modelling, the learned perturbations achieve a remarkable average fooling rate of 80.07%. Note that this is around 8% higher than the best mean fooling rate obtained by an individual network (computed for each row in Tab. 1), which is 72.62% by VGG-19. This again emphasizes the effectiveness of the proposed framework and objectives to simultaneously model perturbations for classifiers with significant architectural differences.

Table 3. Mean fooling rates for 10 perturbations sampled from the distribution of adversaries modelled for multiple target CNNs. The perturbations result an average fooling rate of 80.02% across the 7 target CNNs which is higher than the best mean fooling rate of 72.62% achieved by the generator learned for VGG-19.

Network	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-50	ResNet-152
Fooling rate	83.74	86.94	84.79	73.73	75.24	80.21	75.84

3.6. Black-box attacks for ensemble generator

In this subsection we present the black-box fooling performance of the learned generator for an ensemble of target CNNs. We have learned the ensemble generator G_E with an ensemble of 4 (VGG-F, GoogLeNet, VGG-16, and ResNet-50) target CNNs leaving CaffeNet, VGG-19, and ResNet-152. In Table 4 we report the mean black-box fooling rate (Mean BBFR) obtained by the learned perturbations computed over the three left out models. For comparison, we also present the Mean BBFR achieved by the generators learned for those individual target CNNs computed over the left out 3 models. Owing to the ensemble of targets, generator G_E learns more general perturbations compared to the individual generators and achieves higher fooling rate compared to the individual targets.

Table 4. Generalizability of the perturbations learned by the ensemble generator (G_E).

	G_{VF}	G_G	G_{V16}	G_{R50}	G_E
Mean BBFR	60.63	60.15	71.26	61.87	**76.40**

4. Related Works

Adversarial perturbations [29] have been a tantalizing revelation about machine learning systems. Specifically, the deep neural network based learning systems (*e.g.*, [9, 19, 23]) are also shown to be vulnerable to these structured perturbations. Their ability to generalize to unknown models enables simple ways (*e.g.*, [9]) to launch black-box attacks that fool the deployed systems. Further, existence of image-agnostic perturbations [18, 21, 20] along with their cross model generalizability exposes the weakness of the current day deep learning models. Differing from previous works [18, 21], our work proposes a novel, yet simple and effective objective that enables to learn image-agnostic perturbations. Although the existing objectives successfully craft these perturbations, they do not attempt to capture the space of such perturbations. Unlike the existing works, the proposed method learns a generative model that can capture the space of the image-agnostic perturbations for a given classifier. To the best of our knowledge, the only work which aims to learn a neural network for generating adversarial perturbations via simple feed-forwarding is presented by Baluja *et al.* [2]. They present a neural network which transforms an image into its corresponding adversarial sample. Note that it generates image specific perturbations and doesn't aim to model the distribution like us.

Generative Adversarial Network (GAN): Goodfellow *et al.* [8] and Radford *et al.* [24] have shown that GANs can be trained to learn a data distribution and to generate samples from it. Further image-to-image conditional GANs have led to improved generation quality [12, 33]. Inspired from GAN framework, we have proposed a neural network architecture to model the distribution of universal adversarial perturbations for a given classifier. The discriminator (D) part of the typical GAN is replaced with the trained classifier to be fooled (f). Only the generator (G) part is learned to generate perturbations to fool the discriminator. Also, as we don't have to train D, samples of the target data (i.e., perturbations) are not presented during the training. Through a pair of effective loss functions L_f and L_d, the proposed framework models the perturbations that fool a given classifier.

5. Conclusion

In this paper, we have presented the first ever generative approach to model the distribution of adversarial perturbations for a given CNN classifier. We propose a GAN inspired framework, wherein we successfully train a generator network that captures the unknown target distribution without any training samples from it. The proposed objectives naturally exploit the attributes of the samples (*e.g.*, to be able to fool the target CNN) in order to model their distribution. However, unlike the typical GAN training that deals with a pair of conflicting objectives, our approach has a single well behaved optimization (only G is trained).

The ability of our method to generate perturbations with state-of-the-art fooling rates and surprising cross-model generalizability highlights severe susceptibilities of the current deep learning models. However, the proposed framework to model the distribution of perturbations also enables to conduct formal studies towards building robust systems. For example, Goodfellow *et al.* [9] introduced adversarial training as a means to learn robust models and Tramer *et al.* [30] extended it to ensemble adversarial training, which require a large number of adversarial samples. In addition, the defence becomes more robust if those samples exhibit diversity and allow the model to fully explore the space of adversarial examples. While the existing methods are limited by both generation speed and instance diversity, our method, after modelling, almost instantly produces adversarial perturbations with lots of variety. We have also shown that our approach can efficiently model the perturbations that simultaneously fool multiple deep models.

References

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. In *AAAI conference on Artificial Intelligence*, 2018.

[3] Y. Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1), Jan. 2009.

[4] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013.

[5] B. Biggio, G. Fumera, and F. Roli. Pattern recognition systems under attack: Design issues and research challenges. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(07), 2014.

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), IEEE Symposium on*, 2017.

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems, (NIPS)*, 2014.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of ACM Workshop on Security and Artificial Intelligence*, AISec, 2011.

[12] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 2014.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2013.

[16] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2017.

[17] Y. Liu, X. Chen, C. Liu, and D. X. Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations (ICLR)*, 2017.

[18] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *in The IEEE Conference on Computer Vision and Pattern Recogntion (CVPR)*, 2016.

[20] K. R. Mopuri, A. Ganeshan, and R. V. Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *arXiv preprint arXiv:1801.08092*, 2018.

[21] K. R. Mopuri, U. Garg, and R. V. Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[22] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the Asia Conference on Computer and Communications Security*, 2017.

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, 2016.

[24] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

[26] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for

training GANs. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[29] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[30] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[31] D. Warde-Farley and I. J. Goodfellow. Adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 2016.

[32] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[33] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2017.