AdvGAN++ : Harnessing latent layers for adversary generation

Puneet Mangla[*]
IIT Hyderabad, India
cs17btech11029@iith.ac.in

Surgan Jandial[*]
IIT Hyderabad, India
jandialsurgan@gmail.com

Sakshi Varshney[*]
IIT Hyderabad, India
cs16resch01002@iith.ac.in

Vineeth N Balasubramanian
IIT Hyderabad, India
vineethnb@iith.ac.in

Abstract

Adversarial examples are fabricated examples, indistinguishable from the original image that mislead neural networks and drastically lower their performance. Recently proposed AdvGAN, a GAN based approach, takes input image as a prior for generating adversaries to target a model. In this work, we show how latent features can serve as better priors than input images for adversary generation by proposing AdvGAN++, a version of AdvGAN that achieves higher attack rates than AdvGAN and at the same time generates perceptually realistic images on MNIST and CIFAR-10 datasets.

1. Introduction and Related Work

Deep Neural Networks(DNNs), now have become a common ingredient to solve various tasks dealing with classification, object recognition, segmentation, reinforcement learning, speech recognition etc. However recent works [18, 4, 15, 13, 19, 6] have shown that these DNNs can be easily fooled using carefully fabricated examples that are indistinguishable to original input. Such fabricated examples, knows as adversarial examples mislead the neural networks by drastically changing their latent features, thus affecting their output.

Adversarial attacks are broadly classified into ***White box*** and ***Black box*** attacks. ***White box*** attacks such as FGSM [2] and DeepFool [12] have access to the full target model. In contrary to this *black box* attacks like Carlini and Wagner. [1], the attacker does not have access to the structure or parameters of the target model, it only has access to the labels assigned for the selected input image.

Gradient based attack methods like Fast Gradient Sign Method (FGSM) obtains an optimal max-norm constrained perturbation of

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \tag{1}$$

where J is the cost function and gradient is calculated w.r.t to input example.

Optimization-based methods like Carlini Wagner [1] optimize the adversarial perturbations subject to several constraints. This approach targets L_0, L_2, L_∞ distance metrics for attack purpose. The optimization objective used in the approach makes it slow as it can focus on one perturbation instance at a time.

In contrary to this, AdvGAN [17] used a GAN [3] with an encoder-decoder based generator to generate perceptually more realistic adversarial examples, close to original distribution. The generator network produces adversarial perturbation $G(x)$ when an original image instance (x) is provided as input. The discriminator tries to distinguish adversarial image $(x + G(x))$ with original instance (x). Apart from standard GAN loss, it uses hinge loss to bound the magnitude of maximum perturbation and an adversarial loss to guide the generation of image in adversarial way. Though, AdvGAN is able to generate the realistic examples, it fails to exploit latent features as priors which are shown to be more susceptible to the adversarial perturbations recently [14].

Our Contributions in this work are:

- We show that the latent features serve as a better prior for adversarial generation than the whole input image for the untargeted attacks thereby utilizing the observation from [14] and at same time eliminating the need to follow encoder-decoder based architecture for generator, thus reducing training/inference overhead.
- Since GANs are already found to work well in a conditioned setting [7, 11], we show that we can directly make generator to learn the transition from latent feature space to adversarial image rather than from the whole input image.

[*] Authors contributed equally

In the end, through quantitative and qualitative evaluation we show that our examples look perceptually very similar to the real ones and have higher attack success rates compared to AdvGAN.

2. Methodology

2.1. Problem definition

Given a model M that accurately maps image x sampled from a distribution p_{data} to its corresponding label t, We train a generator G to generate an adversary x_{adv} of image x using its feature map (extracted from a feature extractor) as prior. Mathematically :

$$x_{adv} = G(z|f(x)) \tag{2}$$

such that

$$M(x_{adv}) \neq t, \tag{3}$$

$$\|x - x_{adv}\|_p < \epsilon, \tag{4}$$

where $1 \leq p < \infty, \epsilon > 0$, f represents a feature extractor and ϵ is maximum magnitude $\|.\|_p$ perturbation allowed.

2.2. Harnessing latent features for adversary generation

We now propose our attack, AdvGAN++ which take latent feature map of original image as prior for adversary generation. Figure 1 shows the architecture of our proposed network. It contains the target model M , a a feature extractor f, generator network G and a discriminator network D. The generator G receives feature $f(x)$ of image x and a noise vector z (as a concatenated vector) and generates an adversary x_{adv} corresponding to x. The discriminator D distinguishes the distribution of generator output with actual distribution p_{data}. In order to fool the target model M, generator minimize $M_t(x_{adv})$, which represents the softmax-probability of adversary x_{adv} belonging to class t. To bound the magnitude of perturbation, we also minimize l_2 loss between the adversary x_{adv} and x. The final loss function is expressed as :

$$L(G, D) = L_{GAN} + \alpha L_{adv} + \beta L_{pert} \tag{5}$$

where

$$L_{GAN} = E_{\mathrm{x}}[logD(x) + E_{\mathrm{x}}log(1 - D(G(z|f(x)))], \tag{6}$$

$$L_{adv} = E_{\mathrm{x}}[M_t(G(z|f(x)))], \tag{7}$$

$$L_{pert} = E_{\mathrm{x}}\|x - G(z|f(x))\|_2 \tag{8}$$

Here α , β are hyper-parameters to control the weightage of each objective. The feature $f(.)$ is extracted from one of the intermediate convolutional layers of target model M. By solving the min-max game $arg \min_G \max_D L(G, D)$ we obtain optimal parameters for G and D. The training procedure thus ensures that we learn to generate adversarial images close to input distribution that harness the susceptibility of latent features to adversarial perturbations. Algorithm 1 summarizes the training procedure of AdvGAN++.

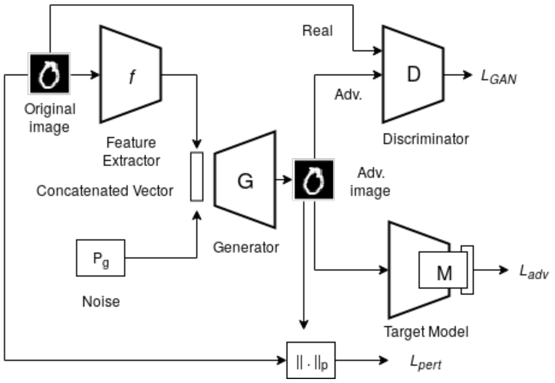

Figure 1: AdvGAN++ architecture.

Algorithm 1: AdvGAN++ training

```
for number of training iterations do
    Sample a mini-batch of $m$ noise samples { $z^{(1)}$, ... $z^{(m)}$ } from noise prior $p_g(z)$ ;

    Sample a mini-batch of $m$ examples {$x^{(1)}$, ... $x^{(m)}$ } from data generating distribution $p_{data}(x)$;

    Extract latent features {$f(x^{(1)})$, ... $f(x^{(m)})$ };

    Update the discriminator by ascending its stochastic gradient. ;
    $\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} log(D(x^{(i)})) + log(1 - D(G(z^{(i)}|f(x^{(i)}))))$;

    Sample a mini-batch of $m$ noise samples { $z^{(1)}$ ,$z^{(2)}$ ... $z^{(m)}$ } from noise prior $p_g(z)$;

    Update the generator by descending its stochastic gradient. ;
    $\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)}|f(x^{(i)})))$
    $+\|x^{(i)} - G(z^{(i)}|f(x^{(i)}))\|_2 +$
    $M_t(G(z^{(i)}|f(x^{(i)})))$
end
```

3. Experiments

In this section we evaluate the performance of AdvGAN++, both quantitatively and qualitatively. We start by describing datasets and model-architectures followed by implementation details and results.

Data	Model	Defense	AdvGAN	AdvGAN++
MNIST	Lenet C	FGSM Adv. training	18.7	**20.02**
		Iter. FGSM training	13.5	**27.31**
		Ensemble training	12.6	**28.01**
CIFAR-10	Resnet-32	FGSM Adv. training	16.03	**29.36**
		Iter. FGSM training	14.32	**32.34**
		Ensemble training	29.47	**34.74**
	Wide-Resnet-34-10	FGSM Adv. training	14.26	**26.12**
		Iter. FGSM training	13.94	**43.2**
		Ensemble training	20.75	**23.54**

Table 1: Attack success rate of Adversarial examples generated AdvGAN++ when target model is under defense.

Data	Target Model	AdvGAN	AdvGAN++
MNIST	Lenet C	97.9	**98.4**
CIFAR-10	Resnet-32	94.7	**97.2**
	Wide-Resnet-34-10	99.3	**99.92**

Table 2: Attack success rate of AdvGAN and AdvGAN++ under no defense

Data	Target Model	Other Model	Attack Success rate
MNIST	LeNet C	LeNet B [16]	20.24
CIFAR-10	Resnet-32 A	Wide-Resnet-34	48.22
	Wide-Resnet-34	Resnet-32	89.4

Table 3: Transferability of adversarial examples generated by AdvGAN++

Datasets and Model Architectures: We perform experiments on MNIST[10] and CIFAR-10[8] datasets wherein we train AdvGAN++ using training set and do evaluations on test set. We follow Lenet architecture C from [16] for MNIST[10] as our target model. For CIFAR-10[8], we show our results on Resnet-32 [5] and Wide-Resnet-34-10 [20].

3.1. Implementation details

We use an encoder and decoder based architecture of discriminator D and generator G respectively. For feature extractor f we use the last convolutional layer of our target model M. Adam optimizer with learning rate 0.01 and β_1 = 0.5 and β_2 = 0.99 is used for optimizing generator and discriminator. We sample the noise vector from a normal distribution and use label smoothing to stabilize the training procedure.

3.2. Results

Attack under no defense We compare the attack success rate of examples generated by AdvGAN and AdvGAN++ on target models without using any defense strategies on them. The results in table 2 shows that with much less training/inference overhead, AdvGAN++ performs better than AdvGAN.

Attack under defense We perform experiment to compare the attack success rate of AdvGAN++ with AdvGAN when target model M is trained using various defense mechanism such as FGSM[2] , iterative FGSM [9] and ensemble adversarial training [16]. For this, we first generate adversarial examples using original model M as target (without any defense) and then evaluate the attack success rate of these adversarial examples on same model, now trained using one of the aforementioned defense strategies. Table 1 shows that AdvGAN++ performs better than the AdvGAN under various defense environment.

Visual results Figure 2 shows the adversarial images generated by AdvGAN++ on MNIST[10] and CIFAR-10[8] datasets. It shows the ability of AdvGAN++ to generate perceptually realistic adversarial images.

Transferability to other models Table 3 shows attack success rate of adversarial examples generated by AdvGAN++ and evaluated on different model $M^{'}$ doing the same task. From the table we can see that the adversaries produced by AdvGAN++ are significantly transferable to other models performing the same task which can also be used to attack a model in a black-box fashion.

4. Conclusion

In our work, we study the gaps left by AdvGAN [17] mainly focusing on the observation [14] that latent features are more prone to alteration by adversarial noise as compared to the input image. This not only reduces training time but also increases attack success rate. This vulnerability of latent features made them a better candidate for being the starting point for generation and allowed us to propose a generator that could directly convert latent features to the adversarial image.

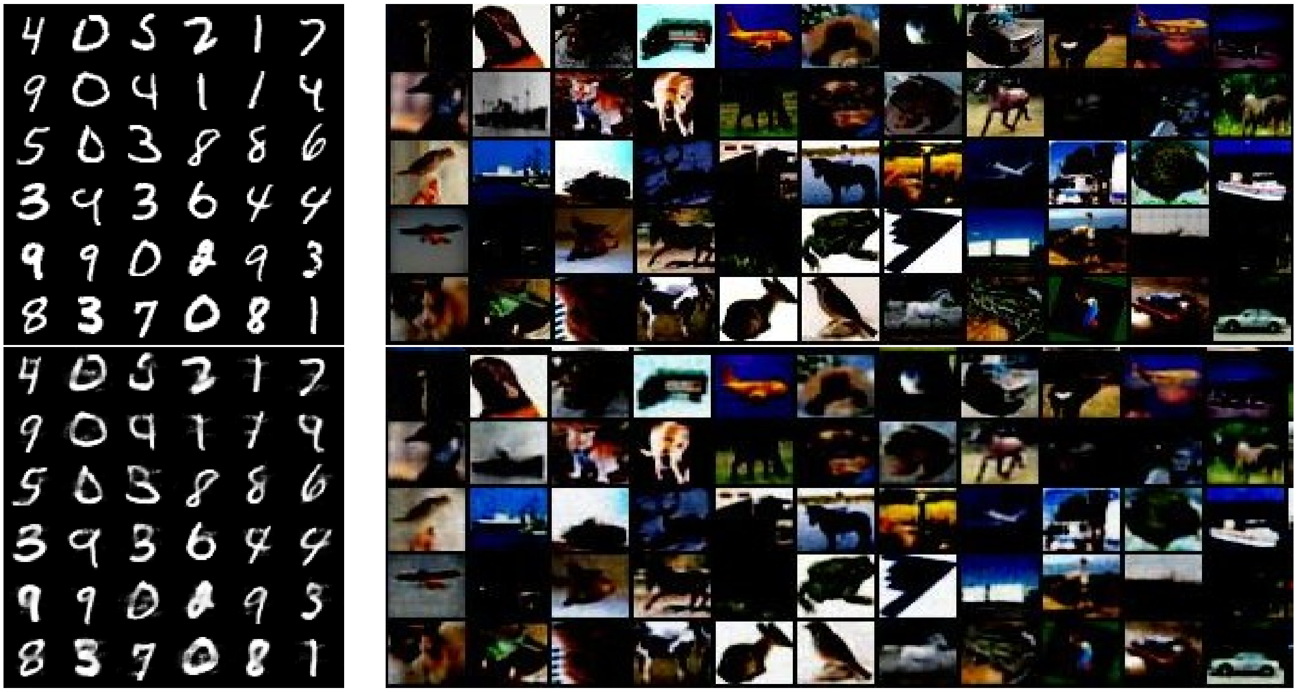

Figure 2: Adversarial images generated by AdvGAN++ for MNIST and CIFAR-10 dataset. Row 1: Original image, Row 2: generated adversarial example.

References

[1] N. Carlini, David, and Wagner. Towards evaluating the robustness of neural networks. *In Security and Privacy (SP), 2017 IEEE Symposium on*, page 3957, 2017. 1

[2] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *In International Conference on LearningRepresentations,*, 2015. 1, 3

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. 1

[4] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. Adversarial examples for malware detection. In S. N. Foley, D. Gollmann, and E. Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 62–79, Cham, 2017. Springer International Publishing. 1

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 3

[6] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *CoRR*, abs/1702.02284, 2017. 1

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2016. 1

[8] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research). 3

[9] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. 3

[10] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. 3

[11] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014. 1

[12] Seyed-Mohsen, Moosavi-Dezfooli, A. Fawzi, and P. Frossard. deepfool: a simple and accurate method to fool deep neural networks,. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),*, 2016. 1

[13] M. Sharif, S. Bhagavatula, L. Bauer, and M. 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*, CCS '16, pages 1528–1540, New York, NY, USA, 2016. ACM. 1

[14] M. Singh, A. Sinha, N. Kumari, H. Machiraju, B. Krishnamurthy, and V. N. Balasubramanian. Harnessing the vulnerability of latent layers in adversarially trained models, 2019. 1, 3

[15] R. Taori, A. Kamsetty, B. Chu, and N. Vemuri. Targeted adversarial examples for black box audio systems. *CoRR*, abs/1805.07820, 2018. 1

[16] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017. 3

[17] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *IJCAI*, 2018. 1, 3

[18] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In *International Conference on Computer Vision*. IEEE, 2017. 1

[19] X. Yuan, P. He, and X. A. Li. Adaptive adversarial attack on scene text recognition. *CoRR*, abs/1807.03326, 2018. 1

[20] S. Zagoruyko and N. Komodakis. Wide residual networks, 2016. 3