Adversarial Attacks on Deep Neural Networks using Generative Perturbation Networks

Sudharman K. Jayaweera
Communications and Information Sciences Laboratory (CISL)
Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM, USA
Email: jayaweera@unm.edu

***Abstract*—A new deep learning (DL) network called Generative Perturbation Network (GPN) is proposed. The GPNs are capable of learning to modify inputs to deep neural network (DNN) models trained to classify images in imperceptible ways leading to misclassification. Unlike previous approaches to generate such adversarial samples to DNN classifiers, the proposed GPN does not need to know the architecture of the neural network that it is trying to attack nor have access to the data used to trained it. It is shown that with just being able to observe the final output labels from the trained classifier to any given input image, the GPNs are able to learn to minimally perturb the images to achieve misclassification. Simulation results show that a proposed GPN can easily degrade the 98.65% accuracy of a trained CNN on the MNIST hand-written digit dataset down to about 52%. Simulation results also show that different regularizations can be used in the generator loss function to achieve various visually desirable characteristics in the generated adversarially perturbed images.**

***Index Terms*—Adversarial attacks, deep learning, generative adversarial networks, generative perturbation networks, machine learning, security, robustness.**

I. Introduction

With the proliferation of deep neural network (DNN) classifiers in various applications, there is the concern regarding their reliability and, in particular, robustness against different types of security threats [1]–[5]. Of particular interest is the DNN classifiers robustness against deliberate adversarial attacks [6], [7]. For example, there have been studies on attacks against trained classifiers that perform content filtering (e.g. SPAM vs. non-SPAM emails, appropriate vs. inappropriate web page contents) in which an adversary generates inputs that will be misclassified resulting in either a webpage with inappropriate contents or a SPAM email pass through the filter [8], [9]. Such misclassifications can result in devastating outcomes in critical applications such as self-driving vehicles. For instance, a self-driving car may rely on a trained DNN image classifier for interpreting traffic signs [2], [10], [11]. Adversarial attacks in which such images are slightly altered at imperceptible levels (to the human eye) yet causing misclassifications by DNNs have already been demonstrated [2]. A misclassification of a STOP sign, for instance, by a self-driving car can clearly lead to fatal outcomes.

Many such adversarial attacks on deep learning (DL) systems implemented on deep networks involve perturbing the inputs in an undetectable way but leading to erroneous responses. Indeed, in [2], [12] the authors discussed how traffic signs can be slightly altered so that a trained image classifier maybe led to provide catastrophically wrong outputs. Intuitively, it can be expected that DNNs indeed will be susceptible to this vulnerability. Typically, they are black-box systems trained on large sets of data. The expectation is that such training will prepare a DNN to accurately classify previously unseen inputs during the test times. This so-called *generalization ability* of a DNN, however, can be difficult to verify. The usual approach of evaluating the performance of a trained DNN over a validation dataset may especially be inadequate against an attacker equipped with its own learning ability. When the space of all possible inputs is large, there is ample room for certain inputs that look almost the same as belonging to a particular class by any distortion measure, or by human perception, to be misclassified by a trained DNN. Apparently, the classifier function learned by the DNN is still highly imprecise over certain parts of the input space that were not sufficiently represented in neither the training nor validation datasets.

There are several commonly encountered adversarial goals in attacking DNNs [6]. In some cases, the goal of the attack might be to slightly change the input to reduce the confidence level of the DNN classifier in its output. This type of attack is called the *confidence reduction*. In *misclassification attacks* the goal is to slightly alter the input so that it is classified to any class other than the correct one. A particular variation of the misclassification attacks, called the *targeted misclassification*, aims to alter the inputs so that they are classified in to a specific target class of interest to the attacker [2], [6]. Another variation of misclassification attacks changes only the specific inputs to be classified into specific output classes and is called the *source-target misclassification* [6]. In all cases, the goal of the adversary is to only alter the inputs ever so-slightly so that although it is misclassified, the distortion introduced by the alteration is as small as possible not to be detected as an altered input by a DNN, any other authentication system or a human inspector.

In this paper, we focus on the general misclassification as the goal of the adversary. As discussed above, there already exist many adversarial attack approaches against classification

systems [1]–[4], [6]. These proposals differ in what they assume to be the capabilities of the attacker. At a very high level, we may make a distinction between attacks that can be performed during training and those that are possible only after the training. In this paper, we assume the latter. The literature on attacks on DNNs during the test time still covers a wide range of attacker capabilities based on what it may have access to [2], [6]:

1) Adversary knows the network architecture and has access to the same training data
2) Adversary knows the network architecture but has no access to the training data
3) Adversary does not know the network architecture but has access to the training data
4) Adversary does not know the network architecture nor has access to the training data but can give inputs to the network and observe the corresponding responses
5) Adversary does not know the network architecture nor has no access to the training data. It cannot present its own inputs to the network but can observe the response of the network to inputs provided by others.

In this paper, our focus is on adversaries belonging to the class 4 above: those who do not know the network architecture nor have access to the training data but can give inputs to the network and observe the corresponding responses. It is worth pointing out that, many existing work on attacks against the DNNs assumes more stronger adversaries belonging to the first three cases above. However, we believe that in most practical situations, an attacker may not know the architecture of a particular DNN classifier that it is attempting to attack. Similarly, there maybe a large number of situations in which the adversary may not have access to the same training data used by the DNN either. As we demonstrate below, however, neither of these stronger capabilities are necessary to effectively render a DNN classification system almost useless.

Summarizing, we pose the following problem: Can we train a adversarial network to learn to perturb an input to an already trained deep network so that the output of the trained network is erroneous but it is unable to detect that the input is perturbed? We specifically consider the image classification with convolutional neural networks (CNNs) and propose a deep network that is geared to learn how best to achieve the above adversarial goal. We term this new type of generative network as a Generative Perturbation Network, or GPN for short.

The rest of the paper is organized as follows: In Section II we briefly introduce the DNN image classification problem. Then, in Section III we describe the proposed adversarial network named the GPN. Next, in Section IV we discuss the adversarial training approach to GPNs with having access to only the final output labels of a DNN that is to be attacked. In Section V we provide results that show the effectiveness of the proposed GPNs in attacking trained classifiers using a digit classification DNN based on the MNIST dataset and, finally, in Section VI we conclude the paper by discussing further work directions.

II. Deep Learning Image Classification Problem

While proposed GPN can be broadly applicable in many applications of deep learning, in the following we formulate its development in the context of DL based image classification. Since [13] won the ImageNet competition using a deep convolutional neural network, they have become the standard approach for machine learning based classifiers. Indeed, over the last decade, deep CNNs have demonstrated the state-of-the-art performance in many image classification problems [14]–[16]. It was reported in [10] that a DNN based approach also won the final phase of the German traffic sign recognition benchmark and was the only method to achieve a better-than-human recognition rate.

The basic structure of a DL image classifier will have several convolutional filter layers followed by few dense neural layers. The input to the network can be either 2 or 3 dimensional depending on grey-scale or color images. The output layer can either provide a final class label or a probability vector of network's confidence.

In this paper, we will specifically focus on the MNIST handwritten digit recognition problem. This dataset is widely accessible and very high classification accuracy can be achieved even with relatively small CNNs [17].

III. Generative Perturbation Networks (GPN)

Consider a deep learning (image) classifier network $\mathcal{C}$ trained to take input $\mathbf{x}$ and classify into one of K possible classes. In general, the last layer of the classifier may considered to be made of K outputs corresponding to the K possible classes where the k-th output of the trained classifier network, denoted by y_k, is given by

$$y_k(\mathbf{x}) = \Pr(\mathbf{x} \text{ belongs to class } k) \tag{1}$$

In this work, we may assume less information than above from the trained classifier that we are planning to attack. Indeed, instead of relative probabilities, we may assume that the only output that an outsider (including the adversaries) may observe is simply the output class label. Hence, we may assume that there is only one output corresponding to a given input $\mathbf{x}$ that an adversary may observe from the classifier $\mathcal{C}$ and it is of the form

$$C(\mathbf{x}) = \underset{1 \le k \le K}{\arg\max}\ y_k(\mathbf{x}) \tag{2}$$

A GPN is made of three component networks (see Fig. 1): A generative network $\mathcal{G}$, a discriminator network $\mathcal{D}$ and a trained classifier network such as the above $\mathcal{C}$. Here, $\mathcal{G}$ and $\mathcal{D}$ are trainable while the classifier $\mathcal{C}$ is an already trained DNN only whose output $C(\mathbf{x})$ to any given input $\mathbf{x}$ can be observed by the rest of the components in the GPN.

Similar to that in a generative adversarial network (GAN), the generator $\mathcal{G}$ in a GPN is used to generate fake inputs. However, there are important distinctions: The generator's goal is not simply to generate any arbitrary fake input that will

confuse the discriminator. Instead, the goal of the generator is to craft a perturbed version of a given input image $\mathbf{x}$ that satisfies two conflicting requirements: The perturbation must be small enough not to be detected while large enough the perturbed version will be classified by $\mathcal{C}$ into a class other than that of the original image. Hence, as shown in Fig. 1, unlike that in a GAN, the input to the generator $\mathcal{G}$ in a GPN includes a reference input $\mathbf{x}$ in addition to the noise input $\mathbf{z}$ [18]. The output of the generator $\mathcal{G}$ is the perturbed input $\hat{\mathbf{x}} = G(\mathbf{x}, \mathbf{z})$. The generator's goal is to generate a perturbed output that is sufficiently close to the reference input $\mathbf{x}$, given as the first element in the input tuple $(\mathbf{x}, \mathbf{z})$, in some sense. Possible closeness criteria may include either being indistinguishable to human inspection or to another machine learning classifier.

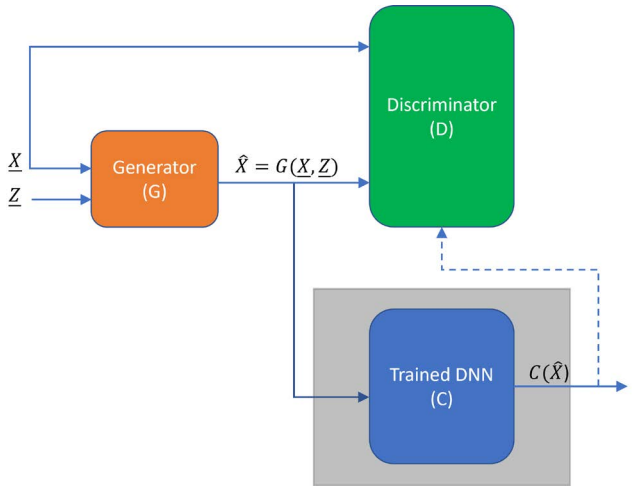


Fig. 1. The basic architecture of a generative perturbation network

The discriminator $\mathcal{D}$ in a GPN also differs from that in a GAN in important ways. First, the input to the discriminator $\mathcal{D}$ is not just the output of the generator as in a GAN. Instead, the input to the discriminator is the pair $(\mathbf{x}, \hat{\mathbf{x}}) = \big(\mathbf{x}, G(\mathbf{x}, \mathbf{z})\big)$. The learning objective of the discriminator in a GPN also differs from that in a GAN. In particular, the goal of the discriminator in a GPN is to compare $\mathbf{x}$ and $\hat{\mathbf{x}}$ in a given input tuple $(\mathbf{x}, \hat{\mathbf{x}})$ and be able to determine whether they belong to the same class or not according to class mapping performed by $\mathcal{C}$. Since we only assume that the adversary's observations are limited to the final output class labels from $\mathcal{C}$, this makes the proposed GPN to work even without fully knowing how many output classes are assumed by $\mathcal{C}$, let alone its internal DNN architecture.

IV. GPN Training

For batch training the discriminator, assume a batch of K inputs $\{\mathbf{x}_k\}_{k=1}^K$. The K batch of reference inputs $\{\mathbf{x}_k\}_{k=1}^K$ are given to the generator along with noise vectors to obtain a K batch of generator outputs $\big\{G\left(\mathbf{x}_k, \mathbf{z}_k\right)\big\}_{k=1}^K$.

Since the goal of the discriminator is to be able to determine whether the two elements in the same input tuple belong to the same class (output is 1) or not (output is 0), it is a binary classifier with a scalar output. For a given input tuple $\big(\mathbf{x}, G(\mathbf{x}, \mathbf{z})\big)$ formed by using the generator's perturbed output corresponding to a reference signal $\mathbf{x}$, the discriminator must learn to determine whether they will be considered to belong to the same class by the trained classifier $\mathcal{C}$. Hence, $D\left(\mathbf{x}, G(\mathbf{x}, \mathbf{z})\right) = 1$ if $C(\mathbf{x}) = C\left(G(\mathbf{x}, \mathbf{z})\right)$ while $D\left(\mathbf{x}, G(\mathbf{x}, \mathbf{z})\right) = 0$ if $C(\mathbf{x}) \neq C\left(G(\mathbf{x}, \mathbf{z})\right)$.

The training input batch to the discriminator is the K batch of tuples given by $\Big\{\big(\mathbf{x}_k, G\left(\mathbf{x}_k, \mathbf{z}_k\right)\big)\Big\}_{k=1}^K$. The corresponding K-vector of labels are denoted by $\mathbf{p}$ whose elements are set to 1 if $C(\mathbf{x}_k) = C\left(G(\mathbf{x}_k, \mathbf{z}_k)\right)$ and 0 otherwise. Observe that, as stated above, this only requires observing the final output from the classifier $\mathcal{C}$ (no need to know the soft outputs). The corresponding K outputs of the discriminator are denoted by the K-vector $\mathbf{q}$.

The objective of the discriminator is to

$$\max_D \ \mathbb{E}\left\{\mathcal{I}_{\{C(\mathbf{X})=C\left(G(\mathbf{X},\mathbf{Z})\right)\}} \log D\left(\mathbf{X}, G(\mathbf{X}, \mathbf{Z})\right)\right\} \tag{3}$$

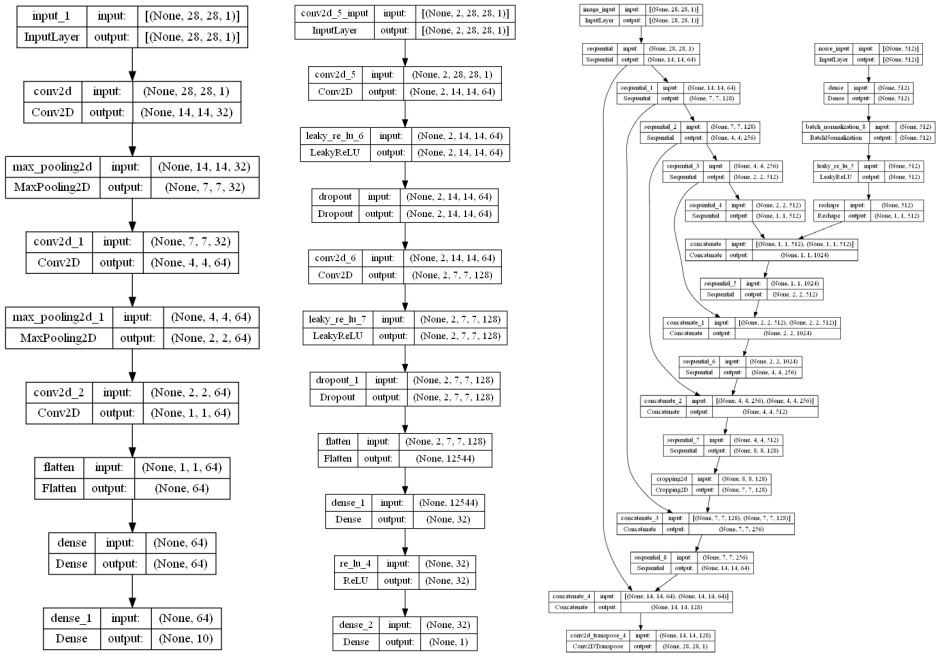
where $\mathbb{E}\{.\}$ is the expectation operator and $\mathcal{I}_E$ is the indicator function of the event E which is equal to 1 if the event E is true and 0 otherwise. Note that, this is equivalent to minimizing the cross-entropy between the desired output $\mathbf{p}$ and the discriminator output $\mathbf{q}$.

On the other hand, the goal of the generator is to craft perturbed outputs $G(\mathbf{x}, \mathbf{z})$ that will simultaneously result in $C\left(\mathbf{x}\right) \neq C\left(G(\mathbf{x}, \mathbf{z})\right)$ and $D\left(\mathbf{x}, G(\mathbf{x}, \mathbf{z})\right) = 1$. Hence, the objective of the generator is to

$$\max_G \ \mathbb{E}\left\{\mathcal{I}_{\{C(\mathbf{X})\neq C\left(G(\mathbf{X},\mathbf{Z})\right)\}} \log D\left(\mathbf{X}, G(\mathbf{X}, \mathbf{Z})\right)\right\} \tag{4}$$

Observe that for $C\left(\mathbf{x}\right) \neq C\left(G(\mathbf{x}, \mathbf{z})\right)$ to be true, the generator's output $G(\mathbf{x}, \mathbf{z})$ needs to be sufficiently different from the reference input $\mathbf{x}$. However, as can be seen from (3), for $D\left(\mathbf{x}, G(\mathbf{x}, \mathbf{z})\right) = 1$ to be true, the generator's output $G(\mathbf{x}, \mathbf{z})$ needs to be sufficiently close to the reference input $\mathbf{x}$ since discriminator is trained to produce this output only when the two elements in the input tuple belong to the same class (according to $\mathcal{C}$). Hence, generator's learning goal in a GPN is made of two conflicting criteria in contrast to that in a GAN [18].

In many cases, an attempt to solve for G and D based on (4) and (3), may not lead to satisfactory results at least in terms of a closeness metric interpreted from the point of view of human visualization of images. Indeed, the generator $\mathcal{G}$ may succeed in crafting images that with very high probability be misclassifed by $\mathcal{C}$ while discriminator can still distinguish them as being sufficiently close to be considered as in the same class as that of the reference input, yet they may look almost random to the human eye. This is possibly due to the fact that the classifier $\mathcal{C}$ is arguably an over-fitted network in the space of all possible input images regardless of how good its (demonstrated) generalization ability on credible images. This, however, can be addressed by modifying the objective function



(a) (b) (c)

Fig. 2. DNN architectures of the component networks of the GPN. (a) classifier (b) discriminator (c) generator

(4) of the generator $\mathcal{G}$ by adding an appropriate regularizing term. In particular, we may modify (4) as

$$\max_{G} \quad \mathbb{E}\left\{\mathcal{I}_{\{C(\mathbf{X})\neq C\left(G(\mathbf{X},\mathbf{Z})\right)\}}\log D\left(\mathbf{X},G(\mathbf{X},\mathbf{Z})\right)\right\} - \lambda L\left(|\mathbf{X}-G(\mathbf{X},\mathbf{Z})|\right) \tag{5}$$

where $\lambda \geq 0$ is a relative weight hyper-parameter and $L(.)$ is a regularizing loss function of the dissimilarity between $\mathbf{x}$ and $G(\mathbf{x},\mathbf{z})$ such as the l_1 and l_2 norms or the Huber loss function [19]. In general, the regularization function must be selected according to the desired definition of the closeness metric.

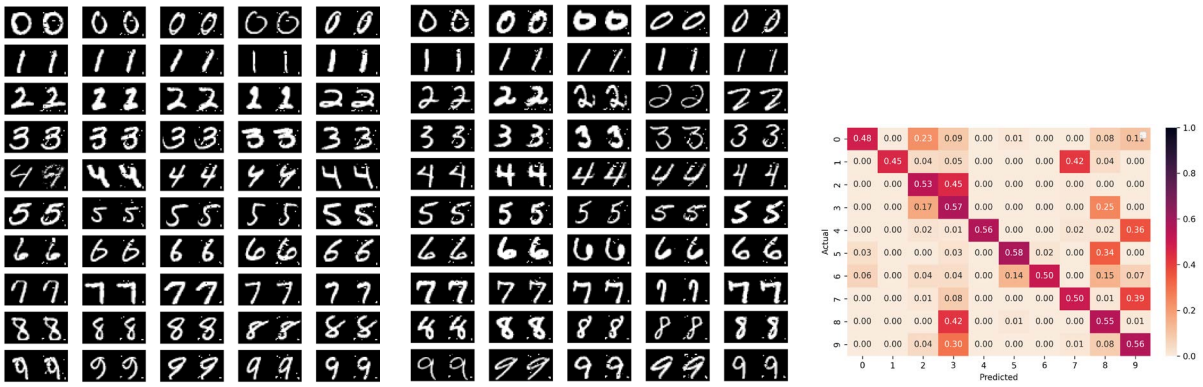
As can be seen from the above discussion, the proposed GPN does not require the knowledge of the architecture of the deep network (i.e. $\mathcal{C}$) that it is trying to attack. Similarly, it does not need to have access to the same training data that was used to train the deep network it is attempting to attack either. Indeed, in training the GPN, the deep network $\mathcal{C}$ is considered purely as a black-box which produces ouputs for a given input $\mathbf{x}$. As we show in the next section, the adversarial learning approach of the GPN is able to simply learn from these observed input-output pairs how to perturb the inputs to the network $\mathcal{C}$ to make it misclassify them.

V. Simulation results

In this section, we describe an experimental setup for designing a GPN that can learn to degrade the performance of a deep neural network trained for hand-written digits classification based on the widely used MNIST dataset [17]. The MNIST dataset contains 60000 training and an additional 10000 testing (grey-scale) images of hand-written digits each of size 28×28. Since we do not want to assume that the GPN have access to the same training data used for training $\mathcal{C}$, we randomly split the 60000 training images of the MNIST dataset into two sets of 30000 each. The first 30000 set is used for training a DNN $\mathcal{C}$ for classifying hand-written digits. The classifier $\mathcal{C}$ is a CNN whose details are shown in Fig.2(a). After training, the classifier $\mathcal{C}$ is tested on the MNIST test dataset made of 10000 images and kept fixed. Without any hyper-parameter tuning, $\mathcal{C}$ showed a 98.65% accuracy on the testing dataset.

Since we do not assume that the GPN has knowledge of the architecture of $\mathcal{C}$, both generator and discriminator were designed to have arbitrary architectures that are independent of that of $\mathcal{C}$ as can be seen, respectively, from Figs 2(b) and 2(c). Note that, the discriminator is a binary classifier and the generator $\mathcal{G}$ is a DNN with two inputs similar to the *conditional adversarial networks* of [20]. The second 30000 MNIST training images of hand-written digits are used to train the discriminator and generator of GPN on how to attack the trained classifier network $\mathcal{C}$.

Table I shows the performance of the discriminator, generator and the classifier on the 10000 test images after GPN training has been completed. As can be seen from the first row of Table I, the GPN has been successful in modifying the test images to reduce the original 98.65% accuracy of the classifier $\mathcal{C}$ down to about 52% when l_1 regularization is used in the generator. Importantly, it is also seen from Table I that the discriminator, on the other hand, is still able to correctly infer that each modified image and the corresponding original image belong to the same class with almost 90% accuracy.



(a) (b) (c)

Fig. 3. Performance of the GPN on MNIST hand-written digit test dataset with l_1 regularization in the generator. (a) Original and GPN-perturbed images that were misclassified by the classifier $\mathcal{C}$ (b) Original and GPN-perturbed images that were not misclassified by the classifier $\mathcal{C}$ (c) Confusion matrix

This provides the evidence that the generated images are not too different from the original images.

TABLE I
PERFORMANCE OF THE GPN ON MNIST TEST DATASET

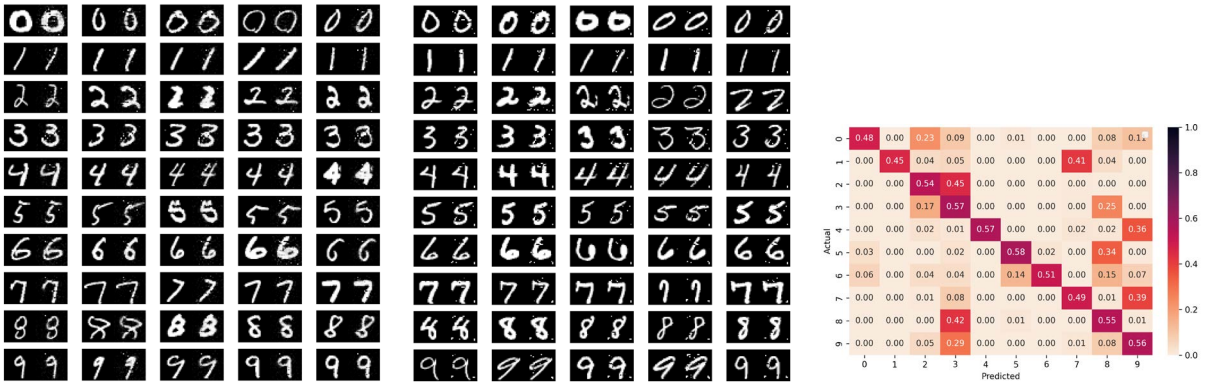
	Discriminator Accuracy	**Classifier Accuracy**
l_1 **regularization**	89.98%	52.63%
Huber regularization	90.80%	56.20%

To verify that the generated perturbed images are indeed sufficiently close to the original images we may use several metrics. First, Fig 3(a) shows original and perturbed images that were misclassified by the classifier $\mathcal{C}$ side-by-side when l_1 regularization is used in the generator. Each row of Fig. 3(a) shows 5 randomly picked examples for a particular digit from misclassified test data. The left and right images in each column correspond to the original and GPN-generated images, respectively. It is interesting to observe from Fig. 3(a) that indeed the perturbations crafted by the generator $\mathcal{G}$ are very small and intuitively meaningful. For example, it has strategically introduced few pixel perturbations in digit 2 that perhaps can possibly be also seen as a 3. It is surprising, however, that the discriminator can still correctly discern that the perturbed image in fact is a digit 2 and not a digit 3 (or any other digit different from digit 2) since its test accuracy is about 89.98%. This hints at the possibility that regardless of how good the performance of the classifier on training and test datasets, it still is an over-fitted network in the space of all possible 28-by-28 pixel grey-scale images.

Similarly, Fig. 3(b) shows the original and perturbed images that were not misclassified by the classifier $\mathcal{C}$ when l_1 regularization is used in the generator. Surprisingly, the perturbations introduced by the GPN in the modified images are not that different from those that are seen in the perturbed images in Fig. 3(a). This supports the belief that indeed the classifier $\mathcal{C}$ is a possibly overfitted network in the space of all possible 28-by-28 pixel grey-scale images. When it is presented with images for a particular digit having illuminated pixels at locations it has possibly not seen before, it seems to almost randomly classify them to one of the other digits whose images most likely had pixel illuminations at those locations. Indeed, the confusion matrix corresponding to this scenario shown in Fig. 3(c) further supports this explanation. For example, when they are misclassifed, the GPN-perturbed digit 6 images are most likely misclassified into digits 8, 5, 9 and 0 in the order of decreasing likelihood.

It is known that there is no universally agreed-upon distortion metric that can be used to capture the image quality perceived by the human eye [21]. In Fig. 4, we have shown the side-by-side comparison of original and GPN-created perturbed images as well as the confusion matrix of the classifier when the Huber loss function is used for regularization in (5). Note also that the summary performance of the discriminator and the classifier with the Huber regularization is also given in the second row of Table I. As can be seen from Table I, there is a slight loss in effectiveness against the classifier $\mathcal{C}$. However, from Fig. 4(a) and 4(b) one may argue that Huber loss seems to give more visually indistinguishable perturbations although this is subject to debate.

In [12], a particular distortion metric was used to demonstrate how small the modifications needed to be to make a trained network misclassify input digits. However, the problem considered in [12] is different from what is considered here and their approach to modifying images renders adopting their definition of distortion for our case inappropriate. Specifically, [12] was interested in generating modified images that will be classified into specific target classes under the much stronger assumption that the attacker knows exactly the architecture of the classifier $\mathcal{C}$. Note that, for that problem one would like to see success rates closer to 1 (as the approach proposed in [12] was indeed shown to achieve) whereas in our problem that is not a realistic nor necessary objective. Any time the classifier $\mathcal{C}$ is made to classify any given digit correctly only with about 50% accuracy, this will render the classifier $\mathcal{C}$ almost useless. Similarly, whenever [12] modified a pixel,



(a) (b) (c)

Fig. 4. Performance of the GPN on MNIST hand-written digit test dataset with Huber regularization in the generator. (a) Original and GPN-perturbed images that were misclassified by the classifier $\mathcal{C}$ (b) Original and GPN-perturbed images that were not misclassified by the classifier $\mathcal{C}$ (c) Confusion matrix

it essentially modified the pixel value to the full brightness of 1. As a result, the distortion metric was defined as the number of pixels that were modified in an input image. Since, in our case the pixel changes are more granular (in steps of $1/256$), a direct comparison of this distortion metric is meaningless. Hence, we define the distortion as the *mean absolute pixel perturbation*. For a coarse comparison, assuming that values of those pixels that were modified were uniformly distributed in $\left\{\frac{0}{256}, \frac{1}{256}, \cdots, \frac{255}{256}\right\}$, we may estimate the possible mean absolute pixel perturbation of [12] to be about $0.0444 \times \frac{257}{2\times256} \approx 0.0223$ among all test images. On the other hand, from our results we compute the mean absolute pixel perturbation of images crafted by our generator $\mathcal{G}$ to be about 0.0299 among all test images. With Huber regularization, these turns out to be about 0.0223 in [12] and 0.0664 in the proposed GPN.

VI. Conclusion

An adversarial deep network named generative perturbation network was proposed to stealthily modify inputs to deep neural network classifiers to degrade their classification accuracy while not being possibly detected as being modified. The proposed GPN has three components: a generator, a discriminator and a trained classifier that is to be attacked. Unlike many previous work, the proposed GPN does not need to know the architecture of the DNN under-attack nor have access to the data that was used to train it. With only being able to observe the output of the trained classifier to a given input, the GPN was shown to craft perturbed images that significantly degrade the classifier's performance. In particular, on the MNIST hand-written digits classification problem, the proposed GPN was able to decrease the performance of a trained classifier with initial test accuracy of about 98% down to only about 52%, without any architectural or hyper-parameter optimization. The immediate further work will include testing the proposed GPN on other image and non-image classification problems, fine-tuning the basic GPN architecture as well as studying the impact of various hyper-parameters on its performance.

References

[1] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 427–436.

[2] A. Aung *et al.*, "Building robust deep neural networks for road sign detection," 12 2017.

[3] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," 06 2018, pp. 1625–1634.

[4] F. Woitschek and G. Schneider, "Physical adversarial attacks on deep neural networks for traffic sign recognition: A feasibility study," in *IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[5] T. Gu *et al.*, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, 02 2019.

[6] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," in *37th IEEE Symposium on Security Privacy*, 2016.

[7] B. Biggio, G. Fumera, and F. Roli, "Pattern recognition systems under attack: Design issues and research challenges," in *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 7, 2014.

[8] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases*, 2013, p. 387–402.

[9] L. Huang *et al.*, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, p. 43–58.

[10] D. Ciresan, U. Meier, and e. a. J. Masci, "Multi-column deep neural network for traffic sign classification," in *Neural Networks*, vol. 32, 2012, p. 333–338.

[11] J. Zhang *et al.*, "Lightweight deep network for traffic sign classification," in *Annals of Telecommunications*, vol. 75, 2020, p. 369–379.

[12] N. Papernot *et al.*, "The limitations of deep learning in adversarial setting," in *IEEE European Symposium on Security and Privacy*, 2016.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, p. 1097–1105.

[14] C. Szegedy *et al.*, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVR)*, 2014, p. 1–9.

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CoRR,abs/1610.02357*, 2016.

[16] K. He *et al.*, "Deep residual learning for image recognition," in *CoRR,abs/1512.03385*, 2015.

[17] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 1998.

[18] I. J. Goodfellow *et al.*, "Generative adversarial networks," 2014.

[19] P. J. Huber, *Robust Statistics*. John Wiley Sons, New York, 1981.

[20] P. Isola *et al.*, "Image-to-image translation with conditional adversarial networks," 2018.

[21] T. M. Cover and J. A. Thomas, *Elements of information Theory*, 2nd ed. Wiley Interscience, 2006.