Generating Adversarial Examples Using Uniform Pixels and Neighbor Aggregation

Glenn Paul P. Gara*†
College of Computer Studies
University of the Immaculate Conception
Davao City, Philippines
ggara@uic.edu.ph

Arnulfo P. Azcarraga
College of Computer Studies
De La Salle University
Manila, Philippines
arnulfo.azcarracaga@dlsu.edu.ph

ABSTRACT

Deep neural networks have gained popularity due to its exceptional performance on several challenging tasks such as image classification, object detection, semantic segmentation, and image generation. Unfortunately, these networks are highly vulnerable to carefully crafted human imperceptible perturbations based on so-called adversarial examples. Adversarial examples have been used to mislead deep neural networks into predicting an incorrect output (i.e. category, label or class), sometimes with high confidence. Most approaches in constructing adversarial inputs require access to the gradients of the network, which is applicable only to gradient-based techniques. Others require only some access to the output function making these methods model-agnostic. In this work, we propose a mechanism that considers the neural network as a black-box by assuming that the network output is observed based only on the examined inputs. We focus our experiments on a model-agnostic adversarial example generation. Specifically, without exploiting the network gradients, we show that by aggregating neighboring images of an input image represented within a low dimensional input space and combine this with a perturbation technique referred to as "uniform pixels", both convolutional neural network and vanilla neural network are vulnerable to incorrect predictions. As such, the proposed method can serve as a basis for designing more robust neural network models.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**;

KEYWORDS

Adversarial example, image perturbation, neighbor aggregation, uniform pixels

ACM Reference Format:
Glenn Paul P. Gara and Arnulfo P. Azcarraga. 2020. Generating Adversarial Examples Using Uniform Pixels and Neighbor Aggregation. In *Proceedings of Philippine Computing Science Congress (PCSC2020).* City of Baguio, Philippines, 7 pages.

*Also with, College of Computer Studies, De La Salle University.
†Also with, Center for Complexity and Emerging Technologies, De La Salle University.

PCSC2020, March 2020, City of Baguio, Philippines

1 INTRODUCTION

Deep neural networks have increased in popularity due to its remarkable performance on challenging tasks such as image classification [5], object detection [8], semantic segmentation [18], image generation [6], etc. Despite the significant success of deep neural networks, literatures have shown that it is susceptible to adversarial examples (or known as adversarial attacks) [4]. These kinds of attacks may cause a deep neural network model to predict incorrectly even if the input images are visually similar from trained samples. The attack is generated by intentionally adding a perturbation, which is imperceptible to human perception. This problem was first introduced by [16], threatening safety-critical systems such as medical diagnosis systems [2] and self-driving cars [14]. In the case of medical image analysis systems, an attack happens when an original image is modified to generate an adversarial image in such a way that the modification is human-imperceptible. Thus, a neural network predicts its class incorrectly even if both the original and the adversarial image are visually identical. In self-driving cars, an attack happens when an attacker creates minimal changes on the stop sign that is human-imperceptible. As a result, the car predicts it as a turn-left sign, which is a decision causing a serious traffic accident. Therefore, it becomes a major concern for a neural network in terms of safety issues.

Adversarial attacks are of two types in terms of the adversary's knowledge: the white-box attack [19], and the black-box attack [1]. White-box attack assumes that the attacker is knowledgeable about everything on the trained neural network model, such as the architecture, hyperparameters, training data, learning algorithm, and model parameters. The attacker can effectively design an image perturbation by leveraging these kinds of knowledge acquired from the model. On the other hand, the black-box attack assumes that the attacker does not have any knowledge about the trained neural network model. The attacker acts only as a standard user knowing only the output of the model. Indeed, one can generate an adversarial attack based solely on the model's result, such as the label and confidence score. There are two ways to craft a perturbation for an adversarial example: the individual attack and universal attack. The individual attack creates a perturbation for every clean input sample. The universal attack, on the other hand, creates a perturbation that is universal for the entire dataset, which means that only one perturbation is crafted and is applied for every clean input sample. Majority of the current attacks generate perturbation independently for each sample. However, it is easy to deploy adversarial examples in a real-world scenario when using universal perturbations [17].

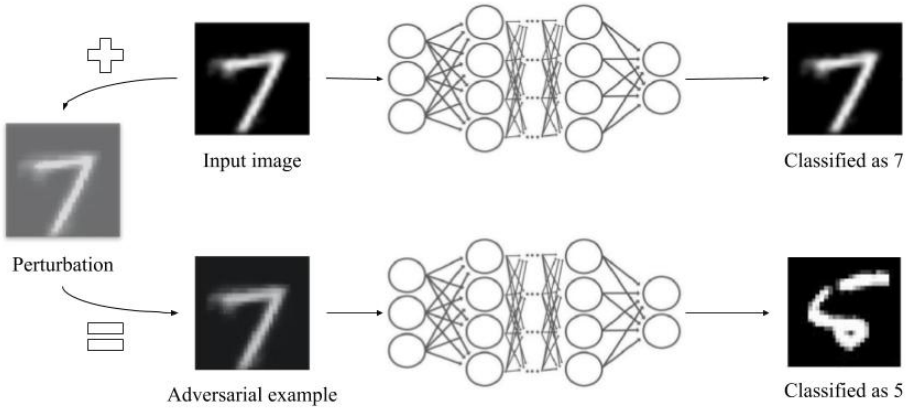

Figure 1: An overview of an attack to a neural network using a generated adversarial example from an input image (original image). Both the input image and adversarial example are visually similar to the character "7", yet the neural network misclassifies the latter as character "5". This means that a neural network is successfully fooled using the perturbed image.

Machine learning researchers proposed significant number of research on adversarial attacks since [16] discuss the vulnerability of neural networks when tested using adversarial examples. An adversarial crafting framework proposed by [13] is divided into two steps, namely the direction sensitivity estimation step and perturbation selection step, which serve as the basis of various proposed adversarial attack methods. One of the known attacks is the Fast Gradient Sign Method (FGSM) [3], which can be done by adding a noise through the loss gradients of the neural network with respect to the input data. The magnitude of the noise is controlled by an ϵ, which is normally a small scalar value. A system called DeepFool [11] studied the decision boundary of a classifier around a certain data point. The authors were trying to look for a patch that will make the data point go beyond the decision boundary for the classifier to give a wrong classification for an input sample. Their results show that DNN classifiers are not robust enough when faced with small perturbations. A black-box attack introduced by [12] utilizes a novel local-search based mechanism to generate a numerical approximation to the gradients of the network, to be used in constructing a small set of pixels to perturb an image. Another method to generate an adversarial example was developed by [10], which aims to build a universal image-agnostic perturbation across training samples. They demonstrated that their universal perturbations can generalize well throughout state-of-the-art deep neural networks. A one-pixel attack proposed by [15] shows that deep neural networks can be fooled where only one pixel value is used to modify an image and which does not require any information about the model.

In this work, we propose a new mechanism to generate adversarial examples by aggregating the neighbors of an input image (test data) from a low dimensional input space, combined with an attack that applies the same value to each feature, referred to as a "uniform pixels" attack. We also examine the robustness of a Convolutional Neural Network (CNN) and a Vanilla Neural Network (VNN) given calculated attacks using the generated adversarial examples.

The rest of the paper is organized as follows. In Section 2, the methods for generating adversarial examples are presented, which includes the neighbor aggregation technique together with a combined attack called "uniform pixels". In Section 3, the experimental set-up and results are presented. In Section 4, we discuss the results and the robustness of CNN and VNN on the generated adversarial examples.

2 METHODS

In this section, we describe how an adversarial attack happens on neural network models. It follows a discussion of two major steps to produce an image perturbation; neighbor aggregation and uniform pixels. Then, we present the proposed method of generating an adversarial example.

2.1 Adversarial attack

An adversarial attack happens when an attacker uses a perturbed image as an input to a machine learning model, crafted intentionally to cause the model to make a mistake in prediction. Fig. 1 shows how an attack happens, particularly in fooling a neural network model to misclassify an image with a human-imperceptible perturbation. The model correctly classifies the original image (input image) from the test images. Using the same set of test image, adversarial attacks (examples) are generated to mislead the neural network model by adding a crafted perturbation. The goal is to make the model predict the adversarial example incorrectly, even if both the original image and the adversarial example are visually similar.

2.2 Neighbor aggregation

Aggregation refers to combining features of multiple samples in order to generate a new sample leveraging the features of other samples. Neighbor aggregation is one of the first steps to generate an image perturbation. As illustrated in Fig. 2, input images (test data) are mapped onto a low dimensional space to easily determine

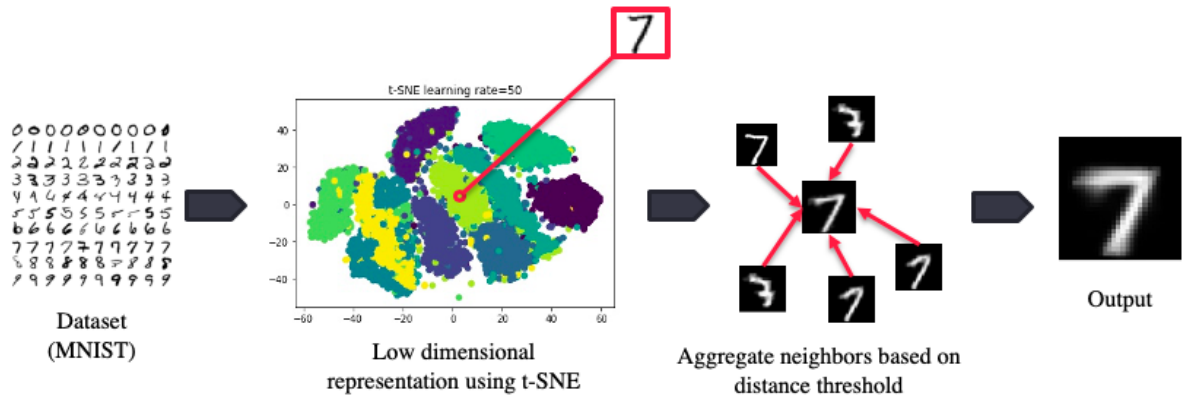

Figure 2: The first step to produce an image perturbation is the neighbor aggregation technique, wherein all input images will be represented in a low dimensional space using t-SNE. Based on the specified distance, all images within the specified distance threshold are merged together using a mean score.

the neighbors of each test image. A t-distributed Stochastic Neighbor Embedding (t-SNE) [9] was used to map test images from a high-dimensional input image space to a two-dimensional space. An attacker can specify a distance threshold to determine which images, within the radius specified by the distance threshold, are going to constitute the set of neighboring images that are to be merged. Merging is done by simply calculating the mean of each pixel. The aggregated image, referred to as output α, is the first step of the generated perturbation. These aggregated images are usually blurry, but with some perceptible pattern that still resembles the original raw test image at the center of the neighborhood.

2.3 Uniform pixels

An attacker sets a set of pixel values (features) having the same value as a second step of generating a perturbation. We call these features as uniform pixels since they have equal or uniform values. Uniform pixels is an *nxn* image with one channel and the same pixel values resembling a blank image. Fig. 3 illustrates the structure of uniform pixels which is set using the notation β. The dimension of uniform pixels must be equal to the dimension of an input sample I and aggregated neighboring images α.

5	5	5	5	5
5	5	5	5	5
5	5	5	5	5
5	5	5	5	5
5	5	5	5	5

Figure 3: An example of 5x5 uniform pixels with a value of $\beta = 5$. The dimension of uniform pixels varies on the dimension of an input sample I.

2.4 Adversarial example generation

The goal is to produce a final image perturbation I' using the neighbor aggregation and uniform pixels mechanisms. As illustrated in Fig. 4, the neighbor aggregation technique and uniform pixels are combined, and adding it to an input image I. An ϵ controls the magnitude of a perturbed image to an image I. Equation 1 is the formula to generate the final output of an adversarial example.

$$I' = I + \epsilon * (\alpha + \beta) \tag{1}$$

Algorithm 1 show the pseudocode of the proposed mechanism. The algorithm expects input images I which are test images and an expected output of perturbed test images. The algorithm initializes the parameters ϵ, d, β, I' where:

- ϵ = real number that is less than or equal to 1, controlling the image perturbation magnitude
- d = an integer which refers to the distance threshold of neighbors to aggregate
- $\beta = m \times m$ array containing the same or uniform values
- I' = an empty list where the perturbed image will be stored

For every iteration of test images, a list of neighbors of I within a low dimensional representation (t-SNE) are identified based on the specified threshold d using the $neighbor()$ function. N holds the set of defined neighboring images. Within the function $aggregate()$, all neighbors are aggregated by calculating the mean score of pixel values. The resulting image from an aggregated neighboring images assigns to α. The image perturbation is the sum of α and β. A perturbation P adds to an image I with a magnitude controlled by ϵ. The resulting adversarial image assigns to I'_j and is stored in I'. Finally, the algorithm outputs all generated adversarial examples, which are test images combined with a crafted image perturbation.

3 EXPERIMENTS

3.1 Neural network architecture

We consider two simple neural networks, Convolutional Neural Network and Vanilla Neural Network (also known as Fully-Connected Network). These two neural networks were trained using the same dataset and used as classifier models for our experiments on the generated adversarial examples. Table 1 and 2 are the specifications of the two neural network architectures. CNN and VNN are both

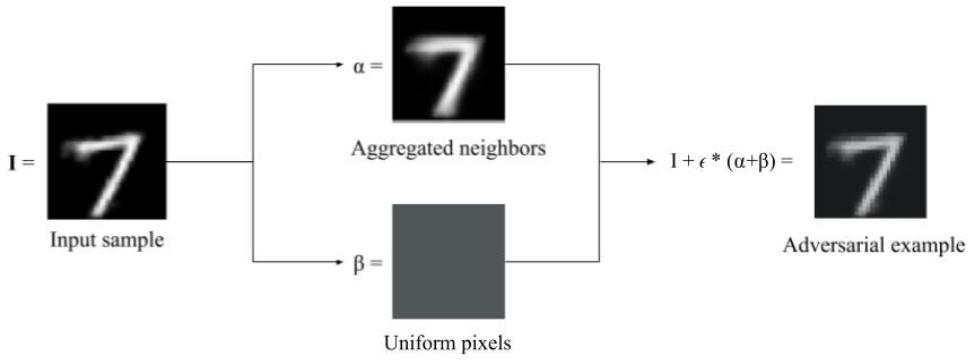

Figure 4: A neighbor aggregation α technique combined with a uniform pixel β mechanism to generate a perturbed image I'.

Algorithm 1: Generating adversarial examples from a testing set via neighbor aggregation and uniform pixels

Input : A finite list $I = [I_1, I_2, \ldots, I_n]$ of test images
Output: A finite list $I' = [I'_1, I'_2, \ldots, I'_n]$ of perturbed images

$\epsilon := i; i \in \mathbb{R} \mid i \leq 1$
$d := k; k \in \mathbb{Z} \mid k \geq 1$
$\beta := m \times m$ uniform values
$I' = [\oslash]$
for *each* I_j *in* I **do**
  $N_j \leftarrow$ `neighbor`(I_j, d)
  `neighbor`(I_j, d) = set of neighboring images I_j within the distance d using t-SNE representation
  $\alpha_j \leftarrow$ `aggregate`(N_j)
  `aggregate`(N_j) = an $m \times m$ mean score pixel values of N_j
  $P_j = \alpha_j + \beta$
  P_j = image perturbation combining neighbor aggregation and uniform pixels for image I_j
  $I'_j \leftarrow I_j + \epsilon * P_j$
  I'_j = perturbed image I_j where ϵ controls the magnitude of P_j
  $I' \leftarrow I'_j$
return I'

Table 1: Vanilla Neural Network architecture for MNIST handwritten digit dataset experiment

Layer (type)	Output Shape	Parameters
FC Linear	(1, 200)	157,000
ReLU	(1, 200)	
FC Linear	(1, 10)	2,010
Softmax	(1, 10)	

trained using the same training parameters (learning rate: 0.001, number of epochs: 5, batch size: 1, optimizer: Adam) both having significant results on test data.

Table 2: Convolutional Neural Network architecture for MNIST handwritten digit dataset experiment

Layer (type)	Output Shape	Parameters
Conv2d	(1, 16, 28, 28)	416
BatchNorm2d	(1, 16, 28, 28)	32
ReLU	(1, 16, 28, 28)	
MaxPool2d	(1, 16, 14, 14)	
Conv2d	(1, 32, 14, 14)	12,832
BatchNorm2d	(1, 32, 14, 14)	64
ReLU	(1, 32, 14, 14)	
MaxPool2d	(1, 32, 7, 7)	
FC Linear	(1, 10)	15,690
Softmax	(1, 10)	

3.2 Dataset

Our experiments used a handwritten digits MNIST dataset [7]. The dataset is famously known for training and testing various machine learning models. The dataset contains 60,000 training images and 10,000 testing images. For this experiment, we used the 10,000 testing images to generate adversarial images and serve as a test set again to determine the robustness of the neural network.

3.3 Neural network models performance on clean dataset

Before testing the trained CNN and VNN on adversarial images, each ensures that they must be able to generalize significantly on a clean dataset. We conducted a test using the test set and based on the results, the test accuracy of CNN and VNN are 0.99 and 0.97, respectively.

4 EXPERIMENTAL RESULTS

Our first experiment is to know the robustness of CNN and VNN by utilizing only a neighbor aggregation technique by gradually increasing the distance threshold d and perturbation magnitude ϵ. Tables 3 and 4 are the results of our experiment for neighbor aggregation without exploiting uniform pixels mechanism. Though the

Table 3: VNN fooling rates (percentage) using neighbor aggregation (without uniform pixels) on increasing distance threshold d and perturbation magnitude ϵ

ϵ	Distance threshold d									
	1	2	3	4	5	6	7	8	9	10
0.1	2.98	2.94	2.94	2.95	2.95	2.96	2.97	2.97	2.97	2.98
0.2	2.96	2.96	2.96	2.96	2.96	2.96	2.96	2.97	2.98	2.98
0.3	2.98	2.97	2.97	2.98	2.98	2.98	2.99	3.0	3.0	3.01
0.4	3.01	3.01	3.02	3.02	3.02	3.03	3.04	3.04	3.05	3.06
0.5	3.06	3.06	3.06	3.07	3.07	3.08	3.09	3.10	3.11	3.12
0.6	3.12	3.12	3.13	3.14	3.14	3.15	3.16	3.17	3.19	3.20
0.7	3.20	3.20	3.20	3.21	3.22	3.23	3.24	3.25	3.27	3.28
0.8	3.28	3.28	3.29	3.30	3.31	3.32	3.33	3.34	3.36	3.37
0.9	3.37	3.37	3.38	3.39	3.40	3.41	3.42	3.44	3.45	3.47
1.0	3.46	3.47	3.47	3.48	3.49	3.50	3.51	3.53	3.55	3.56

Table 4: CNN fooling rates (percentage) using neighbor aggregation (without uniform pixels) on increasing distance threshold d and perturbation magnitude ϵ

ϵ	Distance threshold d									
	1	2	3	4	5	6	7	8	9	10
0.1	1.22	1.23	1.25	1.27	1.28	1.28	1.27	1.27	1.27	1.28
0.2	1.27	1.27	1.27	1.28	1.28	1.28	1.29	1.29	1.29	1.29
0.3	1.30	1.30	1.30	1.31	1.31	1.32	1.33	1.33	1.34	1.34
0.4	1.35	1.35	1.36	1.37	1.38	1.39	1.40	1.40	1.41	1.41
0.5	1.41	1.42	1.43	1.44	1.45	1.47	1.48	1.49	1.49	1.50
0.6	1.50	1.51	1.52	1.54	1.55	1.56	1.57	1.58	1.59	1.59
0.7	1.60	1.61	1.62	1.63	1.65	1.66	1.67	1.68	1.69	1.70
0.8	1.71	1.72	1.73	1.74	1.76	1.77	1.78	1.80	1.81	1.82
0.9	1.83	1.84	1.85	1.86	1.88	1.89	1.90	1.92	1.93	1.94
1.0	1.95	1.96	1.97	1.98	2.0	2.01	2.02	2.04	2.05	2.06

technique can already fool the classifier on some of the generated adversarial images, the misclassification generalization performance is not promising. It appears to have a minimal increase in fooling rates as we increase d and ϵ at the same time. The results also show that VNN is more prone in misclassifying adversarial examples than CNN since the percentage of misclassification results in VNN is a little higher compared to CNN. Based on the results, we consider transforming the image perturbation by adding uniform pixels to our neighbor aggregation technique. Tables 5 and 6 shows the results of adding uniform pixels mechanism. Interestingly, it increases the percentage misclassification results on both models as we increase d and ϵ. For this experiment, we chose uniform pixels $\beta = 5$ due to its promising results. The results show that both VNN and CNN generalizes significantly in misclassifying the generated adversarial examples because of high fooling rate results. Similar to the first experiment, the fooling rate increases as we increase the d and ϵ at once with the highest rate of 86.97 for VNN and 62.70 for CNN. The results suggest that the proposed mechanism can be used to generate an adversarial example to fool a neural network. Figures 5 and 6 shows the generated adversarial examples using

Table 5: VNN fooling rates (percentage) using both neighbor aggregation and uniform pixels ($\beta = 5$) on increasing distance threshold d and perturbation magnitude ϵ

ϵ	Distance threshold d									
	1	2	3	4	5	6	7	8	9	10
0.1	67.66	67.79	67.88	67.94	68.00	68.04	68.10	68.15	68.20	68.24
0.2	69.73	70.98	72.04	72.94	73.73	74.42	75.04	75.59	76.09	76.54
0.3	77.12	77.65	78.13	78.58	79.00	79.38	79.74	80.07	80.38	80.67
0.4	80.96	81.23	81.48	81.72	81.95	82.16	82.36	82.56	82.74	82.91
0.5	83.08	83.23	83.38	83.53	83.66	83.79	83.92	84.04	84.15	84.27
0.6	84.37	84.47	84.57	84.67	84.76	84.85	84.93	85.01	85.09	85.17
0.7	85.24	85.31	85.38	85.45	85.51	85.58	85.64	85.70	85.76	85.81
0.8	85.87	85.92	85.97	86.02	86.07	86.12	86.16	86.21	86.25	86.30
0.9	86.34	86.38	86.42	86.46	86.49	86.53	86.57	86.60	86.64	86.67
1.0	86.70	86.74	86.77	86.80	86.83	86.86	86.89	86.92	86.95	86.97

Table 6: CNN fooling rates (percentage) using both neighbor aggregation and uniform pixels ($\beta = 5$) on increasing distance threshold d and perturbation magnitude ϵ

ϵ	Distance threshold d									
	1	2	3	4	5	6	7	8	9	10
0.1	4.41	4.50	4.56	4.61	4.65	4.69	4.72	4.74	4.76	4.78
0.2	5.59	6.29	6.90	7.44	7.92	8.36	8.75	9.12	9.45	9.75
0.3	10.65	11.47	12.24	12.96	13.63	14.25	14.85	15.40	15.93	16.44
0.4	17.55	18.63	19.67	20.68	21.64	22.55	23.43	24.28	25.10	25.88
0.5	26.95	28.02	29.07	30.09	31.09	32.05	32.99	33.91	34.80	35.66
0.6	36.59	37.52	38.44	39.34	40.21	41.06	41.90	42.70	43.49	44.25
0.7	44.99	45.72	46.43	47.11	47.78	48.43	49.05	49.66	50.25	50.83
0.8	51.38	51.92	52.45	52.97	53.46	53.95	54.42	54.89	55.34	55.77
0.9	56.20	56.62	57.03	57.42	57.81	58.19	58.56	58.92	59.28	59.62
1.0	59.96	60.29	60.62	60.93	61.24	61.55	61.85	62.14	62.42	62.70

the proposed method on VNN and CNN, respectively, with a class prediction result above on each image.

5 CONCLUSION

In this paper, we proposed a new black-box adversarial example generation mechanism to fool a neural network. We created two simple neural network models where we can test our generated adversarial examples. The results show that a neighbor aggregation mechanism combined with a uniform pixel can be used to misclassify our test data effectively. Our experiments show initial results where both models are sensitive to the adversarial examples using the proposed method. However, VNN shows high sensitivity than CNN. The results of our proposed method suggest that it can be used to craft adversarial examples to design extremely robust neural network models.

6 FUTURE WORKS

In this study, we proposed a black-box mechanism to generate adversarial examples to fool neural network models. This study also serves as initial experiments and results on the performance of our proposed method. In our future work, we will conduct several experiments on neighbor aggregation and different uniform pixel

Figure 5: Generated adversarial example of Vanilla Neural Network as we increase the distance d (1 to 10) and perturbation magnitude ϵ (0.1 to 1.0) with a uniform pixels $\beta = 5$. Values above each image are the predicted class labels. The correct class label is 5 as predicted by VNN before applying an image perturbation.

values on neural network models. We are looking forward also to apply our proposed method on state-of-the-art deep learning models and see how it affects its robustness. Also, we will use more benchmark datasets and see how they perform when applied to images with three channels (RGB). A comparison with other black-box adversarial example generation will also be investigated as a follow-up on this study.

REFERENCES

[1] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. 2019. Improving Black-box Adversarial Attacks with a Transfer-based Prior. In *Advances in Neural Information Processing Systems 32*, H Wallach, H Larochelle, A Beygelzimer, F d\textquotesingle Alché-Buc, E Fox, and R Garnett (Eds.). Curran Associates, Inc., 10932–10942. http://papers.nips.cc/paper/9275-improving-black-box-adversarial-attacks-with-a-transfer-based-prior.pdf

[2] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. 2019. Adversarial attacks on medical machine learning. 363, 6433 (2019), 1287–1289.

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

[4] Eric Heim. 2019. Constrained Generative Adversarial Networks for Interactive Image Generation. (4 2019). http://arxiv.org/abs/1904.02526

[5] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and others. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*. 103–112.

[6] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4401–4410.

[7] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[8] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. 2019. CBNet: A Novel Composite Backbone Network Architecture for Object Detection. (9 2019). http://arxiv.org/abs/1909.03625

Figure 6: Generated adversarial example of Convolutional Neural Network as we increase the distance d (1 to 10) and perturbation magnitude ϵ (0.1 to 1.0) with a uniform pixels $\beta = 5$. Values above each image are the predicted class labels. The correct class label is 5 as predicted by CNN before applying an image perturbation.

[9] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

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

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

[12] Nina Narodytska and Shiva Kasiviswanathan. 2017. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 1310–1318.

[13] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 372–387.

[14] Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. 2018. Darts: Deceiving autonomous cars with toxic signs. (2018).

[15] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* (2019).

[16] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. (12 2013). http://arxiv.org/abs/1312.6199

[17] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* (2019).

[18] Yuhui Yuan, Xilin Chen, and Jingdong Wang. 2019. Object-Contextual Representations for Semantic Segmentation. (9 2019). http://arxiv.org/abs/1909.11065

[19] Tianhang Zheng, Changyou Chen, and Kui Ren. 2019. Distributionally adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2253–2260.