2020 16th IEEE International Conference on Automation Science and Engineering (CASE)
August 20-21, 2020, Online Zoom Meeting

Training Artificial Neural Networks by Generalized Likelihood Ratio Method: An Effective Way to Improve Robustness

Li Xiao, Yijie Peng, L. Jeff Hong, Zewu Ke, and Shuhuai Yang

Abstract— **In this work, we proposed a generalized likelihood ratio method capable of training the artificial neural networks with more flexibility: (a)training with discrete activation and loss functions, while the traditional back propagation method cannot train the artificial neural networks with such activations and loss; (b)involving neuronal noises during training and prediction, which will improve the freedom of the model and make it more adaptable to the real environment, especially when environmental noises exist. Numerical results show that the robustness of various artificial neural networks trained by the new method is significantly improved when the input data is affected by both the natural noises and adversarial attacks.**

I. Introduction

Artificial neural network (ANN) has been used as a universal classifier. In recent years, there have been tremendous successes in applying ANNs to image processing, speech recognition, game, and medical diagnosis([1], [2], [3], [4], [5]). In ANN, the inputs such as texts and images are turned into a vector, and each neuron performs a nonlinear transformation on the input vector. A deep learning ANN typically contains multiple layers of convoluted neurons. This complicated machinery maps the input space to the target space. There are synaptic weights in each neuron to be adapted to the surrounding environment based on the loss between the ANN output and target data. The back propagation (BP) method has been the most widely used technique to train ANNs. However, the BP method requires the loss function and activation function to be smooth in ANNs, which limits the capability of ANNs to fit well with the surrounding environments.

Recent work in deep learning has demonstrated that ANNs can be more easily confused by small noises added to the images via snowing, blurring, and pixelation than human being ([6], [7], [8], [9]). Moreover, ANNs are vulnerable to adversarial attacks, where a very small perturbation of the inputs can drastically alter the classification result ([10], [11], [12], [13], [14]). In contrast, the adversarial phenomenon rarely happens for human being ([15]).

Then some interesting questions arise: what are the differences between biological neural networks in the human brains and ANNs? Can we borrow some mechanisms from the biological brain neural networks to improve the robustness of ANNs? There are some noticeable differences between the neurons in human brain and the neurons used in traditional ANNs ([16], [17], [18]). Firstly, the activation of the brain neuron is via an electric impulse, which means the activation function is more like discontinuous rather than Sigmoid or Relu as currently been used. Secondly, human brain perceives an object as a specific category, e.g., dog or cat, which means that the loss function capturing the mechanism of a human brain should be a discontinuous zero-one function, whereas the loss functions in ANNs are smooth, e,g., the cross-entropy function. Furthermore, the brain neuron network is effected directly by the electronic signal sent by a sensory system and the chemical signal from the endocrine, therefore the biological brain functions like learning from the loss value itself rather than the gradient of the loss, which the BP method uses.

In this work, a generalized likelihood ratio (GLR) method is proposed to train ANNs with neuronal noises. Unlike the BP method, GLR trains ANNs directly by the loss value, rather than the gradient of the loss. GLR does not differentiate the sample path of the loss, and it can train ANNs with discontinuous activation and loss functions. Therefore, the new training method generalizes the scope of ANNs to be used in practice, which allows some mechanisms which may more close to what happens in the brain, i.e., (a) learning by the loss value and (b) learning via neurons with discrete activation functions and neuronal noises. The complexity in calculating the GLR estimator is also simpler than the BP method, because there is no need to calculate the backward propagation for the derivatives of the error signals. Furthermore, our method involves neuronal noises during training and prediction, which will improve the freedom of the model and make it more adaptable to the real environment, especially when environmental noises exist.

The GLR method is a recent advance in stochastic gradient estimation studied actively in the area of simulation optimization ([19], [20], [21]). Infinitesimal perturbation analysis (IPA) and the likelihood ratio (LR) method are two classic unbiased stochastic gradient estimation techniques ([22], [23], [24], [25], [26], [27]). IPA allows the parameters in the performance function but requires the continuity (differentiability) of the performance function; LR does not allow the parameters in the performance function, whereas it does not require continuity of the performance function. The GLR method extends two classic methods to a setting allowing both the parameters in the performance function and discontinuous performance function, so that it can be applied to train the parameters in discontinuous ANNs.

We test the classification results of various trained ANNs

Li Xiao is with Institute of Computing Technology, Chinese Academy of Science, Beijing, 100871 China (xiaoli@ict.ac.cn). Yijie Peng and Shuhuai Yang are with the Department of Management Science and Information Systems, Guanghua School of Management, Peking University, Beijing, 100871 China (pengyijie@pku.edu.cn, 1801214092@pku.edu.cn). L. Jeff Hong is with the Department of Management Science, School of Management, Fudan University, Shanghai, 200433, China (hong_liu@fudan.edu.cn).

when the input data is corrupted by both the natural noises and the adversarial attacks. The robustness of all ANNs trained by the GLR method is significantly improved compared with the ANN with the Sigmoid activation function and cross-entropy loss function trained by the BP method. Especially, the GLR method can bring as high as 20% performance improvement under adversarial attacks. It can also bring as high as 13% performance improvement with corruption noises attack, and the improvement can be further increased to 21% when we use Laplacian instead of Gaussian as neuronal noises.

II. Method

A. Setup and Background

Suppose we have inputs $\vec{X}^{(1)}(n) := (x_1^{(1)}(n), .., x_{m_1}^{(1)}(n))$, $n = 1, \ldots, N$. For the nth input, the ith output of the tth level of neurons is given by

$$v_i^{(t)}(n) := \sum_{j=0}^{m_t} \theta_{i,j}^{(t)} x_j^{(t)}(n) + r_i^{(t)}(n),$$
$$x_i^{(t+1)}(n) := \varphi\left(v_i^{(t)}(n)\right), \quad i = 1, \ldots, m_{t+1} ,$$

where $x_j^{(t)}(n)$ is the jth input of the tth level of neurons (jth output of the $(t-1)$th level of neurons), $\theta_{i,j}^{(t)}$ is a synaptic weight, $r_i^{(t)}(n)$ is a noise with density function $f_{i,t}(r_i^{(t)}(n))$, $v_i^{(t)}(n)$ is the ith signal, and φ is the activation function. The synaptic weights $\theta_{i,j}^{(t)}$, $j = 0, \ldots, m_t$, are the parameters to be trained in the ANN. It is required that $x_0^{(t)}(n) \equiv 1$, and $\theta_{i,0}^{(t)}$ is called a bias. Figure 1 illustrates the structure of a neuron in the ANN.

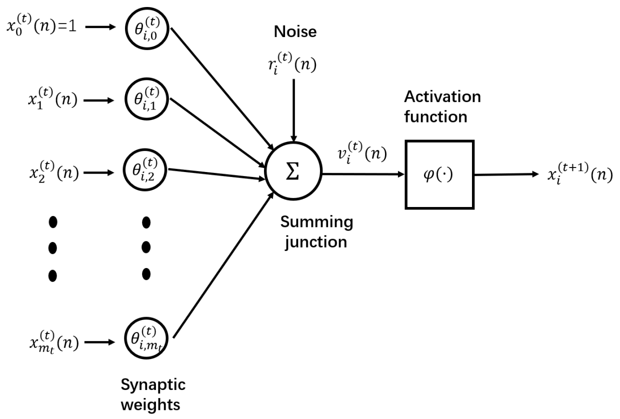

Fig. 1. Structure of a neuron.

The classic ANN does not include the noise, i.e., $r_i^{(t)}(n) \equiv 0$, and the ANN considered in our work generalizes the classic one by adding a (random) noise to the neurons. The activation function φ is a nonlinear function. The Sigmoid function is a popular activation function defined by

$$\varphi_s(v) := 1/(1 + \exp(-sv)),$$

where $s > 0$ is a constant. The Sigmoid function is smooth. Notice that with parameter s increasing to infinity, the Sigmoid function converges to a threshold function, i.e.,

$$\lim_{s\to\infty} \varphi_s(v) = \varphi_o(v) := \begin{cases} 1 & \text{if } v > 0, \\ 0 & \text{if } v < 0. \end{cases}$$

In Figure 2, we can see the curves of the Sigmoid functions with different parameters and the threshold function.

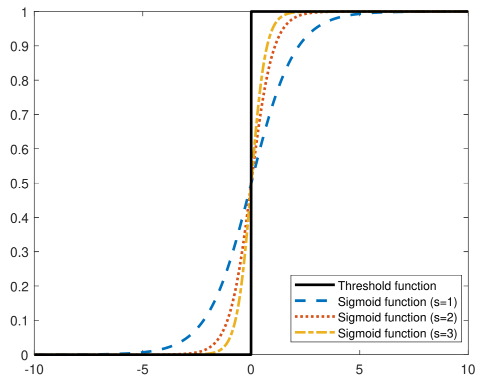

Fig. 2. Threshold and Sigmoid activation functions.

Suppose the ANN has τ levels of neurons, and $\vec{X}^{(\tau)}(n) := (x_1^{(\tau)}(n), \ldots, x_{m_\tau}^{(\tau)}(n))$ is the output vector of the ANN given the nth input data. Let $\vec{O}(n) := (o_1(n), \ldots, o_{m_\tau}(n))$ be the real observation vector given the nth input data, and $L(\vec{X}^{(\tau)}(n), \vec{O}(n))$ be a loss function of the outputs of ANN and observations. In classification, a popular loss function is the cross-entropy loss function given by

$$L_c(\vec{X}^{(\tau)}(n), \vec{O}(n)) = -\sum_{i=1}^{m_\tau} o_i(n) \log\left(p_i(\vec{X}^{(\tau)}(n))\right),$$

where

$$p_i(\vec{X}^{(\tau)}(n)) := \frac{\exp\left(x_i^{(\tau)}(n)\right)}{\sum_{j=1}^{m_\tau} \exp\left(x_j^{(\tau)}(n)\right)}, \quad i = 1, \ldots, m_\tau .$$

The functions p_i, $i = 1, \ldots, m_\tau$, are called softmax functions. Note that the cross-entropy loss function is smooth. Alternatively, we can also use the following $0-1$ loss function:

$$L_o(\vec{X}^{(\tau)}(n), \vec{O}(n)) =$$
$$\mathbf{1}\left\{\max_{i=1,\ldots,m_\tau} p_i(\vec{X}^{(\tau)}(n)) = \max_{i=1,\ldots,m_\tau} o_i(\vec{X}^{(\tau)}(n))\right\} .$$

To train the ANN, we want to minimize the expected loss:

$$\mathcal{E}(\theta) = \mathbb{E}\left[L(\vec{X}^{(\tau)}(n), \vec{O}(n))\right]$$

where θ is a vector containing all synaptic weights. To solve the optimization, the stochastic approximation (SA) ([28]) is applied

$$\theta(n) = \theta(n-1) - \lambda_n G(n) , \qquad (1)$$

where λ_n is the learning rate and $G(n)$ is an unbiased stochastic gradient estimator of $\mathcal{E}(\theta)$, i.e.,

$$\mathbb{E}[G(n)] = \nabla_\theta \mathcal{E}(\theta)|_{\theta=\theta(n-1)} \ . \tag{2}$$

The BP method is the most popular stochastic gradient estimator ([29]):

$$B_{a,b}^{(l)}(n) := \delta_a^{(l+1)}(n) x_b^{(l)}(n),$$

where $B_{a,b}^{(l)}(n)$ is an unbiased stochastic derivative estimator with respect to synaptic weight $\theta_{a,b}^{(l)}$, and

$$\delta_i^{(t)}(n) := \begin{cases} e_i^{(\tau)}(n)\varphi'\left(v_i^{(\tau-1)}(n)\right), & \text{if } t=\tau, \\ \varphi'\left(v_i^{(t-1)}(n)\right)\left(\sum_{j=1}^{m_t}\theta_{j,i}^{(t)}\delta_j^{(t+1)}(n)\right), & \text{if } t<\tau, \end{cases} \tag{3}$$

with the error signal $e_i^{(\tau)}(n)$ defined by

$$e_i^{(\tau)}(n) := \frac{\partial L(\vec{X}^\tau(n), \vec{O}(n))}{\partial x_i^{(\tau)}(n)} \ .$$

B. Generalized Likelihood Ratio method

BP directly differentiates the sample path of the output and it is shown to be a special form of IPA in the journal version archived in Researchgate with DOI: 10.2139/ssrn.3318847, so it requires the sample path of the output is Lipchitz continuous and differentiable almost surely to ensure unbiasedness in (2). Therefore, BP cannot deal with the stochastic gradient estimation for ANN with discontinuous activation function and loss function. ANN used in our work may contain a threshold activation function, which leads to a discontinuous sample path of the output. In this work, we derive a GLR estimator under a special case with the Sigmoid activation and a bounded loss function with a bounded gradient. The derivation shows the connection between the GLR method and the BP method. We construct an ANN with the Sigmoid activation function:

$$u_i^{(t)}(n) := \sum_{j=0}^{m_t}\theta_{i,j}^{(t)} y_j^{(t)}(n) + r_i^{(t)}(n)$$
$$y_i^{(t+1)}(n) := \varphi_s\left(u_i^{(t)}(n)\right), \quad i=1,\dots,m_{t+1},$$

and an ANN with the threshold activation function:

$$\eta_i^{(t)}(n) := \sum_{j=0}^{m_t}\theta_{i,j}^{(t)} z_j^{(t)}(n) + r_i^{(t)}(n)$$
$$z_i^{(t+1)}(n) := \varphi_o\left(\eta_i^{(t)}(n)\right), \quad i=1,\dots,m_{t+1} \ .$$

Theorem 1: Assuming that density function $f_{a,l}(\cdot)$ of the noise $r_a^l(n)$ (added to the a-th output of the $(l-1)$-th level of neurons) is differentiable and loss function $L(\cdot)$ is bounded and with a bounded gradient w.r.t. $\vec{X}^{(\tau)}(n)$, we have

$$\frac{\partial}{\partial\theta_{a,b}^{(l)}}\mathbb{E}\left[L(\vec{Z}(n),\vec{O}(n))\right]$$
$$=\mathbb{E}\left[-L(\vec{Z}(n),\vec{O}(n))\ z_b^{(l)}(n)\frac{\partial \log f_{a,l}(r_a^{(l)}(n))}{\partial r_a^{(l)}(n)}\right] \ .$$

Remark 1: The proof can be found in the journal version archived in Researchgate with DOI: 10.2139/ssrn.3318847. We can show that for an ANN with certain smoothness in activation and loss functions,

$$B_{a,b}^{(l)}(n) = \frac{\partial L(\vec{X}^{(\tau)}(n), \vec{O}(n))}{\partial\theta_{a,b}^{(l)}} \ .$$

The GLR estimator is defined by

$$L_{a,b}^{(l)}(n) := L(\vec{X}(n),\vec{O}(n))\ \omega_{a,b}^{(l)}(n), \tag{4}$$

where

$$\omega_{a,b}^{(l)}(n) := -x_b^{(l)}(n)\frac{\partial\log f_{a,l}(r_a^{(l)}(n))}{\partial r_a^{(l)}(n)} \ .$$

From the proof of Theorem 1, we can see that for an ANN under certain regularity conditions, the estimator of the BP algorithm and the GLR estimator can be linked via integration by parts. For an ANN with a threshold activation function, the GLR estimator is derived by first smoothing the threshold activation function, which becomes the Sigmoid function, then integration by parts, and last taking limit to retrieve the threshold activation in the derivative estimator. These three components have also been used to derive the GLR method in a general framework [21], where we can find the smoothing technique is applied to a general discontinuous sample performance function without actually explicitly constructing the smoothing function. The GLR method can be generalized to deal with stochastic gradient estimation or even higher order gradient for the ANN with more general discontinuous activation and loss functions.

Remark 2: The BP method differentiates the loss and transmits the error signal from the output layer backward throughout the entire ANN via the chain rule of the derivative, whereas in Eq.4, the GLR method does not differentiate the loss and directly uses the loss function scaled by a weight function, which can be viewed as an interaction between the interior mechanism of ANN and the loss in a surrounding environment, to train the ANN.

Remark 3: The BP method is computationally efficient because it only requires simulating a forward function propagation and backward error propagation for once, and the derivatives w.r.t. all synaptic weights $\theta_{i,j}^{(t)}$, $j=1,\dots,m_t$, $i=1,\dots,m_{t+1}$, $t=1,\dots,\tau-1$, are estimated. The GLR method is even faster than BP, since its computation only contains one forward function propagation for estimating the derivatives w.r.t. all parameters. The analysis on computational complexity of the BP and GLR methods can be found in the journal version archived in Researchgate with DOI: 10.2139/ssrn.3318847. For a Gaussian random noise $r_i^{(t)}$ with zero mean and variance $\sigma_{i,t}^2$, we have

$$\frac{\partial\log f_{i,t}(r_i^{(t)})}{\partial r_i^{(t)}} = -\frac{r_i^{(t)}}{\sigma_{i,t}^2} \ .$$

C. Implementation Details

In implementation, we add a Gaussian noise with zero mean and certain variance to each neuron. Then, the GLR gradient estimator used in Eq.4 for a synaptic weight associated with the signal from the b-th neuron at the l-th level to the a-th neuron at the $(l+1)$-th level is

$$\frac{L(\vec{X}(n),\vec{O}(n))x_b^{(l)}(n)r_a^{(l)}}{\sigma_{a,l}^2}\,. \tag{5}$$

Here $x_b^{(l)}$ is the signal from the b-th neuron at the l-th level and $r_a^{(l)}$ is the noise of the a-th neuron at the $(l+1)$-th level, $L(\vec{X}(n),\vec{O}(n))$ is the loss. For simplicity, we choose a common variance σ^2 for the noises in all neurons. The training procedure by the GLR method is summarized in ***Algorithm 1***.

Algorithm 1: **Setup:** Input $\vec{X}^{(1)}(n)$, observations $\vec{O}(n)$, and variance σ^2.

Step One: Calculate loss output $L(\vec{X}(n),\vec{O}(n))$ and the GLR gradient $G(n)$ in Eq.2 via Eq.5.

Iterations: Replicate the above procedure K times to generate i.i.d. gradient estimates $G_1(n), G_2(n), ..., G_K(n)$.

Output: An average of GLR gradient estimates $\frac{1}{K}\sum_{i=1}^{K} G_i(n)$, which is used in Eq.1 for updating parameter.

III. Experiments

A. Dataset Preparation and Network Structure

We test the performance of the GLR method for training ANNs in the example of identifying the numbers from 0-9 of the MNIST dataset, where there are 10000 images in total. The images are split into the training set and testing set in a 6:4 ratio. Each image is resized to be a 14×14-pixels vector for facilitating the training. The appearance of the images is shown in Figure.3.

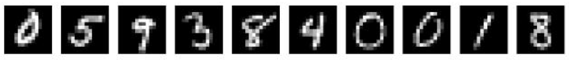

Fig. 3. Images of the numbers in the MNIST dataset.

The ANN to be trained in the experiments have three layers: an input layer, a hidden layer, and an output layer. The structure of the ANN is depicted in Figure.4(a). The dimension of the input layer is 196, the same as the size of the image. The hidden layer has 20 neurons, and the output layer has 10 neurons representing 10 numbers. The integer value of the label needs to be converted into a 10-unit array as the target of the ANN. For example, when the label is 2, the target vector should be $[0,0,1,0,0,0,0,0,0,0]$. The operations between the layers are illustrated in Figure.1. The inputs of the input layer and hidden layer first go through linear operations with Gaussian noises added on and then nonlinear activation functions are operated. The bias term in our ANN is set to be 1 at the head of each input array.

B. Training Procedure

The number of replications in ***Algorithm 1*** is set as $K=10000$. We apply the SA in Eq.1 with mini-batches and the batch size is set as 25, which takes about 12 seconds to run

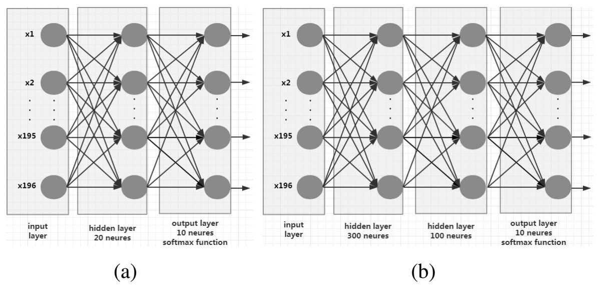

Fig. 4. (a) Structure of ANN trained by the GLR method; (b) Structure of ANN for generating adversarial samples with two hidden layers.

in python in a desktop with Intel i7-6700 CPU @ 3.40 GHz for each iteration. Each epoch contains 1680 iterations which takes about five hours to run. The step size is set as 0.1 and the noise variance is set as $\sigma^2 = 4$. In Figure.5, we show the training and validation errors of ANNs with the Sigmoid and threshold activation functions (as plotted in Figure.2) in the MNIST dataset, and the errors of ANNs converge fast after 12 epochs (≈ 20000 iterations).

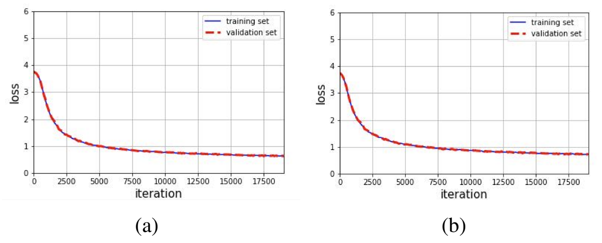

Fig. 5. Losses in the training and validation sets of a single hidden layer ANN trained by GLR with activation functions as Sigmoid in (a) and threshold in (b), respectively.

C. Robustness to Adversarial Attacks

1) Adversarial Samples: We generate the adversarial samples by an ANN with two hidden layers (as depicted in (b) of Figure.4). The ANNs are trained and validated by the BP method. The limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS)([10]) and fast gradient sign method (FGSM)([11]) are used to generate the adversarial samples of 5000 images randomly chosen from the testing set. The adversarial samples of the images generated by FGSM are shown in Figure.6.

2) Adversarial Test: Adversarial test is performed on several single hidden layer ANNs (Figure.4(a)) with different activation and loss functions. The accuracy is measured by the percentage of correct predictions over all adversarial samples. The ANN with the same structure in (a) of Figure.4 trained by BP is used as the baseline for comparisons. Table.I presents the results when adversarial samples are generated by the ANN with two hidden layers (Figure.4(b)). The accuracies of the prediction on the original samples and the adversarial samples generated by the aforementioned two methods are reported.

Besides the Sigmoid and threshold activation functions, three other discontinuous activation functions are also used

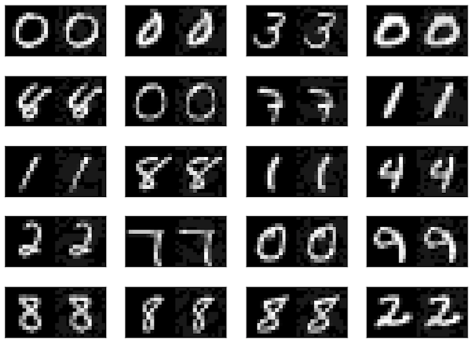

Fig. 6. Samples of the paired images of the original samples (left) and adversarial samples (right). The adversarial samples are generated by the FGSM.

Activations + Entropy	**Orig**	**Adv_L_BFGS**	**Adv_FGSM**
Sigmoid (trained by BP)	0.96	0.57	0.28
Sigmoid	0.94	0.77	0.45
Threshold	0.93	0.73	0.52
$y = \|x\|$	0.94	0.78	0.53
Activations + 0-1 loss	**Orig**	**Adv_L_BFGS**	**Adv_FGSM**
Sigmoid	0.84	0.76	0.58
Threshold	0.83	0.72	0.57

TABLE I

ADVERSARIAL TESTS FOR ANNS WITH DIFFERENT ACTIVATION AND LOSS FUNCTIONS TRAINED BY GLR. THE ADVERSARIAL SAMPLES ARE GENERATED BY AN ANN WITH TWO HIDDEN LAYERS. **ORIG** MEANS THE ACCURACY TESTED ON ORIGINAL SAMPLES. **ADV_L_BFGS** MEANS THE ACCURACY TESTED ON SAMPLES GENERATED BY THE L_BFGS METHOD. **ADV_FGSM** MEANS THE ACCURACY TESTED ON SAMPLES GENERATED BY FGSM.

in the experiments. Each test runs for 12 epochs and all of them demonstrate high accuracies on predicting the original samples after training. We also test the performance of an ANN with 0-1 loss function (the loss is 0 for a correct prediction and 1 otherwise), trained by the GLR. The ANN with 0-1 loss converges slower than the classic cross entropy loss (see Table.I), so we run 24 epochs in training.

In Tables.I , an ANN with one hidden layer trained by the BP method reaches an accuracy of 0.96 in predicting the original samples. However, the accuracy of the same ANN reduces dramatically to 0.57 and 0.28 in predicting the adversarial samples generated by L_BFGS and FGSM, respectively. This substantiates an observation in literature, i.e., the same adversarial samples can effectively attack ANNs under different architectures.

All ANNs with the cross-entropy loss function trained by the GLR method achieve accuracies in predicting original samples comparable to the ANN trained by the BP method (above 93% in accuracy). Notice that GLR can train the ANNs with discontinuous activation functions, e.g., threshold function, and discontinuous loss functions, e.g, 0-1 loss, which cannot be handled by BP. Moreover, the ANNs trained by the GLR method have much higher accuracies (about 20% increase) in predicting adversarial samples compared to the ANN trained by the BP method. Another interesting observation is that although the ANNs with 0-1 loss only reach accuracies less than 90% in predicting original samples, they might lead to even higher accuracies in predicting the adversarial samples than the ANNs with the cross-entropy loss.

D. *Robustness to Natural Noises*

Different from adversarial attack where the input images are affected by small, additive, classifier-tailored perturbations, natural noises add small, general, classifier-agnostic perturbations to the input images. In [6], an IMAGENET-C benchmark generated from IMAGENET [30] offers various corruption types with five severity levels for each type. In this work, we apply four algorithms to generate the corrupted samples for the MINST dataset. Assume the accuracy of the corruption type c at the severity level $s(1 \leq s \leq 5)$ for model f is defined as $Acc^f_{s,c}$, and then the average accuracy is defined as the evaluation metrics:

$$Acc^f_c = \frac{1}{5}\sum_{s=1}^{5} Acc^f_{s,c} \ . \tag{6}$$

	Sigmoid (trained by BP)	Sigmoid	Sigmoid (0-1 loss)
Orig	0.96	0.94	0.84
Gaussian	0.64	0.75	0.72
Impulse	0.40	0.51	0.55
Glass	0.38	0.44	0.44
Contrast	0.25	0.45	0.46
Average	0.418	0.538	0.543
	$y = \|x\|$	Threshold	Threshold (with 0-1 loss)
Orig	0.94	0.93	0.83
Gaussian	0.74	0.75	0.72
Impulse	0.51	0.51	0.51
Glass	0.42	0.44	0.43
Contrast	0.38	0.39	0.41
Average	0.513	0.523	0.518

TABLE II

TEST ROBUSTNESS TO NATURAL NOISES FOR ANNS WITH DIFFERENT ACTIVATION AND LOSS FUNCTIONS TRAINED BY GLR. FOUR TYPES OF CORRUPTION NOISES ARE ADOPTED AND THE AVERAGE ACCURACY IS COMPUTED.

We compute the average accuracy under four types of natural noises. The images corrupted by the Gaussian noises under five levels of severity are shown in Figure 7. In Table.II, the ANN trained by the GLR method has a better performance than that trained by the BP method. Although the performance of the ANN with 0-1 loss is worse than that with a cross-entropy loss, an ANN with 0-1 loss achieves the best performance in predicting the images corrupted by the natural noises.

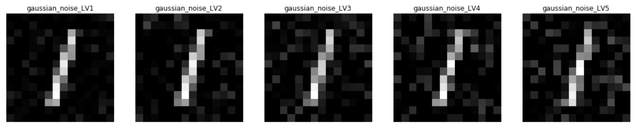

Fig. 7. A sample corrupted by different levels of Gaussian noises.

IV. Conclusions

In this work, a GLR method is proposed for training ANNs with neuronal noises. Unlike the classic BP method, the GLR trains ANNs directly by the loss value rather than the gradient of loss and can handle ANNs with discontinuous activation and loss functions because it does not differentiate the loss output. Therefore, the GLR method could be a powerful tool to explore some brain-like learning mechanisms which allow more freedom to better represent the surrounding environment. The robustness of all ANNs trained by the GLR method is significantly improved compared with the ANN with the Sigmoid activation function and cross-entropy loss function trained by the BP method, which indicates that the new training method is a very promising tool for enhancing the security of ANNs used in practice.

Other direction lies in reducing the variance of the stochastic gradient estimation for ANNs and speed up the training procedure, so that our method can be used in the deep learning ANNs with higher complexity. Adding regularization functions to the loss function for further improving robustness also deserves future research.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 71901003, 71571048, 71720107003, 71690232, 91846301, and 71790615, and by CAS Pioneer Hundred Talents Program (2017-074).

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[2] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. Q. Nelson, J. Mega, and D. Webster, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, 2016.

[5] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, pp. 115–, 2017.

[6] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *CoRR*, vol. abs/1807.01697, 2018. [Online]. Available: http://arxiv.org/abs/1807.01697

[7] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do CIFAR-10 classifiers generalize to cifar-10?" *CoRR*, vol. abs/1806.00451, 2018. [Online]. Available: http://arxiv.org/abs/1806.00451

[8] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *CoRR*, vol. abs/1805.12177, 2018. [Online]. Available: http://arxiv.org/abs/1805.12177

[9] H. Dan, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," *NeurIPS*, 2018.

[10] I. S. J. B. D. E. I. G. C. Szegedy, W. Zaremba and R. Fergus, "Intriguing properties of neural networks," *In International Conference on Learning Representations*, 2014.

[11] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv*, vol. preprint arXiv:1412.6572, 2014.

[12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *In Security and Privacy (SP), 2017 IEEE Symposium on*, p. 39–57, 2017.

[13] A. F. Seyed-Mohsen Moosavi-Dezfooli and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 2574–2582, 2016.

[14] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," *NeurIPS*, 2016.

[15] G. F. Elsayed, S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. J. Goodfellow, and J. Sohl-Dickstein, "Adversarial examples that fool both human and computer vision," *CoRR*, vol. abs/1802.08195, 2018.

[16] P. Dayan and L. Abbott, *Theoretical Neuroscience*. The MIT Press, 2018.

[17] D. Hebb, *The Organization of Behavior*. The Wiley and Sons, 1949.

[18] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, 1943.

[19] S. Asmussen and P. W. Glynn, *Stochastic Simulation: Algorithms and Analysis*. Springer Science & Business Media, 2007, vol. 57.

[20] M. C. Fu, "Stochastic gradient estimation," in *Fu, Michael C. (ed.), Chapter 5 in Handbooks of Simulation Optimization*. Springer, 2015, pp. 105–147.

[21] Y. Peng, M. C. Fu, J.-Q. Hu, and B. Heidergott, "A new unbiased stochastic derivative estimator for discontinuous sample performances with structural parameters," *Operations Research*, vol. 66, no. 2, pp. 487–499, 2018.

[22] Y.-C. Ho and X.-R. Cao, *Discrete Event Dynamic Systems and Perturbation Analysis*. Kluwer Academic Publishers, Boston, MA, 1991.

[23] P. Glasserman, *Gradient Estimation via Perturbation Analysis*. Kluwer Academic Publishers, Boston, 1991.

[24] L. J. Hong, "Estimating quantile sensitivities," *Operations Research*, vol. 57, no. 1, pp. 118–130, 2009.

[25] R. Y. Rubinstein and A. Shapiro, *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. Wiley, New York, 1993.

[26] G. C. Pflug, *Optimization of Stochastic Models*. Kluwer Academic, Boston, 1996.

[27] B. Heidergott and H. Leahu, "Weak differentiability of product measures," *Mathematics of Operations Research*, vol. 35, no. 1, pp. 27–51, 2010.

[28] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003.

[29] S. S. Haykin, *Neural Networks and Learning Machines*. Pearson Upper Saddle River, NJ, USA:, 2009, vol. 3.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.