Deep Robust Neural Networks Inspired by Human Cognitive Bias Against Transfer-based Attacks

Yuuki Ogasawara(✉), Masao Kubo, and Hiroshi Sato

Department of Computer Science, National Defense Academy of Japan, Yokosuka, Japan
ogayuukinda@gmail.com, {masaok,hsato}@nda.ac.jp
http://www.nda.ac.jp/~masaok, http://www.nda.ac.jp/~hsato

Abstract. In recent years, with the proliferation of cloud services, the threat of Transfer-based attacks, a type of Adversarial attacks, has increased. Adversarial Training is known as an effective defense against this attack, but it has been pointed out that it degrades accuracy against normal data and robustness against random noise(Gaussian noise). To solve these problems, we focus on the human visual function, which has robustness while maintaining high accuracy. The contribution of top-down processing, in which the feedforward signal is overwritten by some bias factor, has been pointed out as the reason for this. From this perspective, we propose a new model based on Neural Networks using human cognitive bias. This is an algorithm that overwrites signals according to human cognitive bias and is expected to reproduce human visual functions. Evaluation experiments on two different datasets suggest that the proposed model is robust against Transfer-based attacks. Furthermore, the proposed model can mitigate the accuracy degradation of the normal data to a limited extent, suggesting that it is robust against random noise.

Keywords: Adversarial attacks · Adversarial examples · Transfer-based attacks · Random noise · Cognitive bias · Neural networks · Robustness

1 Introduction

Currently, there is a social trend to rely on AI services for critical decision-making, such as translation services, object detection services, authentication systems, and automated driving. Along with this social foundation of AI services, Adversarial attacks [3] that exploit these services are becoming more prevalent. In particular, attacks that aim to degrade the functionality of AI services by creating modified data (Adversarial Examples) as input to Machine Learning (ML) models that lead to incorrect results have become a social problem. Transfer-based attacks [6,15,17] are well-known as one of these attacks, and Adversarial

Y. Kambayashi et al. (Eds.): AICON 2022, LNICST 477, pp. 72–85, 2023.
https://doi.org/10.1007/978-3-031-29126-5_6

Training (AT) [1,12,21] is a defensive method. Transfer-based attacks are attacks in which the attacker creates adversarial examples from predicted models (Surrogate models) and induces misclassification of the ML models, even if the internal information of the ML models are unknown. AT is a method to prevent incorrect results by adding adversarial examples to the ML models training data. Although AT is widely used, it is known to 1) degrade pure service performance (Clean accuracy), and 2) be overly sensitive to random noise [23]. Therefore, it is necessary to develop robust models that can replace AT while suppressing these side effects.

Our goal is to improve "Adversarial robustness" (robustness against Transfer-based attacks) and "Noise robustness" (robustness against random noise) with as little degradation of Clean accuracy as possible. To achieve this goal, we focus on the property that human vision mechanism has Adversarial robustness [7] and Noise robustness [10], while high accuracy rate of about 95% [18]. Computer vision is fully feedforward neural networks and is unidirectional in nature. On the other hand, human vision has feedback functions during feedforward, which means that the signal is overwritten or modified by experience and knowledge. This modification function is believed to reduce the effects of adversarial examples and random noise, allowing for a more essential visual representation [7,10]. Inspired by these findings on vision, we focus on a model called loosely symmetric neural networks (LSNN), which overrides feedforward signals with human cognitive biases. We also developed a new model (LS-DNN), and proposed its application to ML models.

The contributions of this paper are threefold.

(I) LS-DNN has the potential to suppress the range of degradation of Clean accuracy, which is a side effect of Adversarial Training (AT).
(II) It is suggested that LS-DNN has Adversarial robustness.
(III) It is suggested that LS-DNN has Noise robustness.

2 Related Work

2.1 Adversarial Examples

Given adversarial examples x_{adv}, small perturbation $\| \delta \|_p \in \mathbb{R}^d$, original input $x \in \mathbb{R}^d$, correct label of x $t \in \mathbb{R}^K$, trained parameters θ. We use $f_\theta(x) : \mathbb{R}^d \to \mathbb{R}^K$ to be the ML models function. The prediction class $k(x)$ of x becomes (1).

$$k(x) = \underset{k=1,2,\cdots,K}{\operatorname{argmax}} f_{\theta_k}(x) \tag{1}$$

Adversarial examples can be defined by (2) and the task of finding δ to maximize the loss function J, under the constraint that $k(x_{adv}) \neq k(x)$ and $\| \delta \|_p$ must not be exceeded ε. The optimal $\delta(= \delta^*)$ is given by (3).

$$x_{adv} := x + \delta^* \tag{2}$$

$$\delta^* := \underset{\delta}{\text{argmax}}\, J(\theta, x_{adv}, t) \quad s.t.\ \parallel \delta \parallel_p \leq \varepsilon,\ k(x_{adv}) \neq k(x),\ p = \{0, 1, 2, \infty\} \quad (3)$$

Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) are well known as representative quadrature methods. These are introduced in this paper.

Fast Gradient Sign Method. Goodfellow et al. [12] proposed the Fast Gradient Sign Method (FGSM), which can generate adversarial examples simply and quickly. FGSM computes the sign of the direction in which the loss function $J(\theta, x, t)$ is maximum and adds ε. (see (4)). Here, ε is a hyperparameter that adjusts the magnitude of the perturbation, and the sing function is the function that determines the sign and is (5). Adding this perturbation δ^* to the input x generates adversarial examples x_{adv}.

$$x_{adv} = x + \delta^* = x + \varepsilon \cdot sign(\nabla_x J(\theta, x, t)) \quad (4)$$

$$sign(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x = 0) \\ -1 & (x < 0) \end{cases} \quad (5)$$

Projected Gradient Descent. Madry et al. [1] proposed Projected Gradient Descent (PGD), an improved method of FGSM. This improvement makes it possible to misclassify with fewer perturbations than FGSM since the perturbations are less likely to be noticed by the defender. PGD iteratively updates the perturbation αsmaller than ε every n steps by (6). Where $x + \mathcal{S}$ denotes that the norm from the input image is a region within ε.

$$x^{n+1} = \mathit{\Pi}_{x+\mathcal{S}}(x^n + \alpha \cdot sign(\nabla_x J(\theta, x^n, t))) \quad (6)$$

2.2 Transfer-Based Attacks

One of the most common attacks using adversarial examples is known as Transfer-based attacks [3,20] characterized by a property called "Transferability" [6,17]. Transferability is the property of being affected by adversarial examples between different ML models performing similar tasks. We will refer to the ML models under attack as "Target models" and ML models predicted by the attacker as "Surrogate models". Paternot et al. [16] showed that Target models in the cloud provided by Google and Amazon can be misclassified by Surrogate models. In these cloud services, the internal information of Target models is often unknown, and they only return query results. The attack procedure is described below. 1) The attacker gathers a variety of information in advance. 2) The attacker selects (or creates) and trains Surrogate models based on the gathered information. 3) The attacker creates adversarial examples using learned Surrogate models. 4) The attacker attacks Target models with the adversarial

examples to induce misclassification. Thus, Black-box attacks are realistic scenarios, and defensive measures against them are urgent issues. The basic strategy of defensive measures is to construct models with properties that are as immune as possible to the influence of adversarial examples.

2.3 Defense Against Transfer-Based Attacks

The defense methods against Transfer-based attacks can be roughly classified into two categories: attack detection and model robustness. For attack detection, it is pointed out that attack and detection evasion methods are powerful and not sufficiently effective [14]. For the robustness of the model itself, methods that introduce ensembles or use non-differentiable activation functions [4] have been proposed. However, the effects of both of these methods are limited and sufficient robustness has not been achieved.

Adversarial Training (AT). Among the many defense methods, AT is known as the most effective method. Therefore, we use AT as a benchmark against LS-DNN. AT is a method that dynamically generates adversarial examples during training and adds them to the regular data. In other words, the goal is to learn θ to minimize the expected value of the Adversarial error by (7). Where $\mathcal{D}$ denotes the data distribution over which (x, t) pairs exist, $\mathbb{E}$ the expected value, and $\mathcal{S}$ the allowed l_p norm.

$$\min_{\theta}\{\mathbb{E}_{(x,t)\sim\mathcal{D}}[\max_{\delta\in\mathcal{S}} J(\theta, x_{adv}, t)]\} \tag{7}$$

Goodfellow et al. [12] propose a method to introduce a hyperparameter α that adjusts the ratio of adversarial examples by (8).

$$\tilde{J}(\theta, x, t) = \alpha \cdot J(\theta, x_{adv}, t) + (1 - \alpha) \cdot J(\theta, x, t) \tag{8}$$

Problems of Adversarial Training. AT has some problems, and the following 3 negative points are pointed out in this paper. 1) Degrading Clean accuracy. This is because there is a trade-off [11,21] between Adversarial robustness and Clean accuracy. 2) The performance of Adversarial robustness depends on hyperparameters such as Surrogate models architecture [2,20], the algorithm of the adversarial examples [20,21], α in (8). In other words, if the adversarial examples generated by the attacker differs from the hyperparameters used in the AT, the Adversarial robustness performance may be degraded. 3) It is not robust against random noise. Senzaki et al. [23] reported that ML models trained with AT tend to be less accurate against random noise. Random noise is generated by (9) and is evaluated by Random accuracy.

$$x_{rnd} = x + \delta = x + \zeta \cdot sign(\{r_i\}_{i=1\ldots n}) \tag{9}$$

where x_{rnd} is the input with random noise, r_i is a variable following a normal distribution $\mathcal{N}(0, 1)$, n is a variable of x. ζ is a hyperparameter that determines the intensity of the random noise and $sign$ function is represented by (5).

3 Proposed Method

3.1 Loosely Symmetric Neural Networks

Taniguchi et al. [9] proposed Loosely Symmetric Neural Networks (LSNN), which is a model of human cognitive bias applied to Neural Networks [5]. LSNN is an attempt to reproduce human physiological causal relationships observed between neurons using the LS model [19]. LSNN architecture consists of three layers: input layer, hidden layer, and output layer, and a logistic sigmoid function is used as the activation function of hidden layer. Now we have Neural Networks with k layers, and the total input to the j node in the k layer (output layer) is x_j^k, and the output of this node is y_j^k. The parameter from the i node in the $k-1$ layer to the j node in the k layer is $\theta_{i,j}^{k-1,k}$. The number of nodes in the $k-1$ layer is n. The output of each node is expressed by (10).

$$x_j^k = \sum_{i=1}^{n} \theta_{i,j}^{k-1,k} \cdot y_i^{k-1} \quad , \qquad y_j^k = \frac{1}{1+e^{-x_j^k}} \tag{10}$$

When the number of nodes in the k layer is m, the loss function J between the output value y_i^k and the true value t_i is MSE, J is expressed by (11). Where δ_i^k is the difference between the network output and the true value.

$$J = \frac{1}{2}\sum_{i=1}^{m}(y_i^k - t_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\delta_i^k)^2 \tag{11}$$

This loss function J grows in proportion to the square of the difference between the output and true values, Backpropagation is the process of updating the value of the parameter $\theta_{i,j}^{k-1,k}$ so that it decreases. When the learning rate is α, the change in the parameter $\Delta\theta_{i,j}^{k-1,k}$ is calculated by (12).

$$\Delta\theta_{i,j}^{k-1,k} = -\alpha \cdot \delta_j^k \cdot y_j^k(1-y_j^k) \cdot y_i^{k-1} \tag{12}$$

Consider a situation where event p is a cause and event q is a result. Based on the causal relationship between nodes x_j^k and y_i^{k-1}, indicators a, b, c, and d in (Table 1) represent the degree of activity and inactivity of the nodes to which the LS model is applied.

Table 1. Contingency table of the LS model.

p (*cause*)	q (*result*)	
	q (x_j^k *is activated*)	$\neg q$ (x_j^k *is not activated*)
p (y_i^{k-1} *is activated*)	$a = y_i^{k-1}$	$b = 1 - y_i^{k-1}$
$\neg p$ (y_i^{k-1} *is not activated*)	$c = 1 - x_j^k$	$d = x_j^k$

Focusing on the hidden layer node y_i^{k-1} to be overwritten, the conditional probability $P(q|p)$ can be expressed by (13).

$$a = y_i^{k-1} = P(q|p) = \frac{a}{a+b} \tag{13}$$

By adding the bias terms $\frac{bd}{b+d}$ and $\frac{ac}{a+c}$ to the denominator and numerator of $P(q|p)$, LSNN realizes the signal overwriting process in (14).

$$LS(q|p) = LS(y_i^{k-1}) = \frac{a + \frac{bd}{b+d}}{a+b+\frac{ac}{a+c}+\frac{bd}{b+d}} \tag{14}$$

Therefore, the amount of change in the parameters can be expressed by (15).

$$\Delta_{LS}\theta_{i,j}^{k-1,k} = -\alpha \cdot \delta_j^k \cdot y_j^k(1-y_j^k) \cdot LS(y_i^{k-1}) \tag{15}$$

Thus, the procedure for updating the parameters of the LSNN is as follows, and its architecture is shown in (Fig. 1).

(I) Compute output layer nodes up to the k layer by Feedforward.
(II) Use causality at the k and $k-1$ nodes, and overwrite the value of the $k-1$ node by (14).
(III) Using the values of the hidden layer nodes overwritten by (II), the parameters are updated by Backpropagation.

The procedure (II) is henceforth referred to as the LS Module.

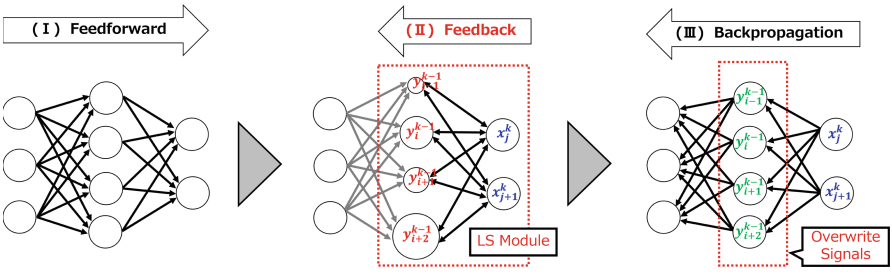


Fig. 1. LSNN architecture.

3.2 LS-DNN Development

We propose LS-DNN (Loosely Symmetric Deep Neural Networks) with additional hidden layers of neural networks and arbitrary number of LS Module applied. The purpose is to determine to what extent the number, position, and combination of LS Module applied to improve robustness and suppress Clean accuracy. In other words, to gain insight into the tendency of cognitive bias to work effectively or counterproductively. In this paper, we have created a simple all-coupled DNN model with up to 5 layers ($k = 5$). Thus, the procedure for updating the parameters of the LS-DNN is as follows, and its architecture is shown in (Fig. 2).

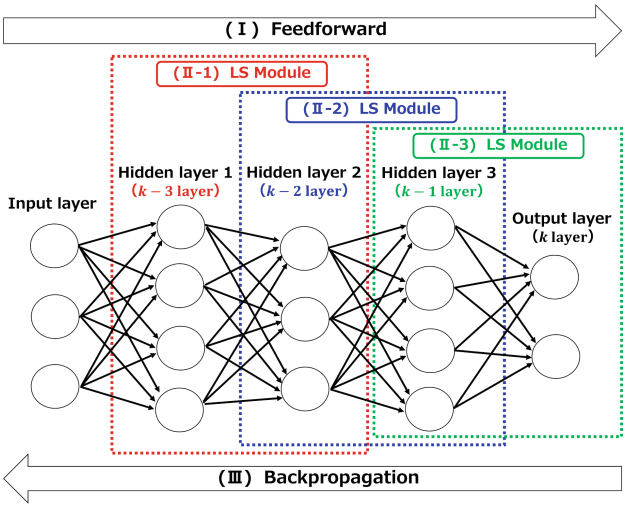


Fig. 2. LS-DNN architecture.

(I) Feedforward computes the output of nodes up to the k layer (output layer).
(II) Select any hidden layer ($k-3$, $k-2$, $k-1$) and overwrite the values of all hidden nodes by the LS Module, starting from the hidden layer closest to the input layer.
(III) Using the values of the hidden layer nodes overwritten with (II), update the parameters by Backpropagation.

The selection of an arbitrary hidden layer is explained here. LS-DNN are modeled according to the number and location of hidden layers where LS Modules are introduced. For example, for $k = 5$, LS-DNN is classified into 7 different models. An example of the notation is given below. LS(12) represents two LS Modules (red and blue dotted boxes) in the first and second hidden layers.

4 Experiments and Discussion

4.1 Methods

Evaluation experiments are based on Papernot et al.'s proposed method [16]. Experimental scenarios assume that an accurate dataset has already been obtained through prior information gathering, but information on the Target models are unknown. We used MNIST dataset [22] and Fashion-MNIST dataset [8]. These are well-known datasets for 10-class classification tasks. 60,000 training data and 10,000 test data are pre-populated with the correct answers labeled. The computer server used in the experiments is Ubuntu 20.04 OS, Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GH, and 503GB memory. LS-DNN is evaluated in three experiments. 1) Accuracy against normal data(Clean accuracy)

and Training time. 2) Accuracy against adversarial examples(Adversarial accuracy). 3) Accuracy against random noise(Random accuracy). As for 1), LS-DNN is expected to be computationally expensive due to feedback structure. For service providers, the additional computation time for training is a cost to develop and maintain. Therefore, we also evaluate the training time. (Fig. 3) shows images of the 1st experiment(Exp-1), the 2nd experiment(Exp-2), and the 3rd experiment(Exp-3).

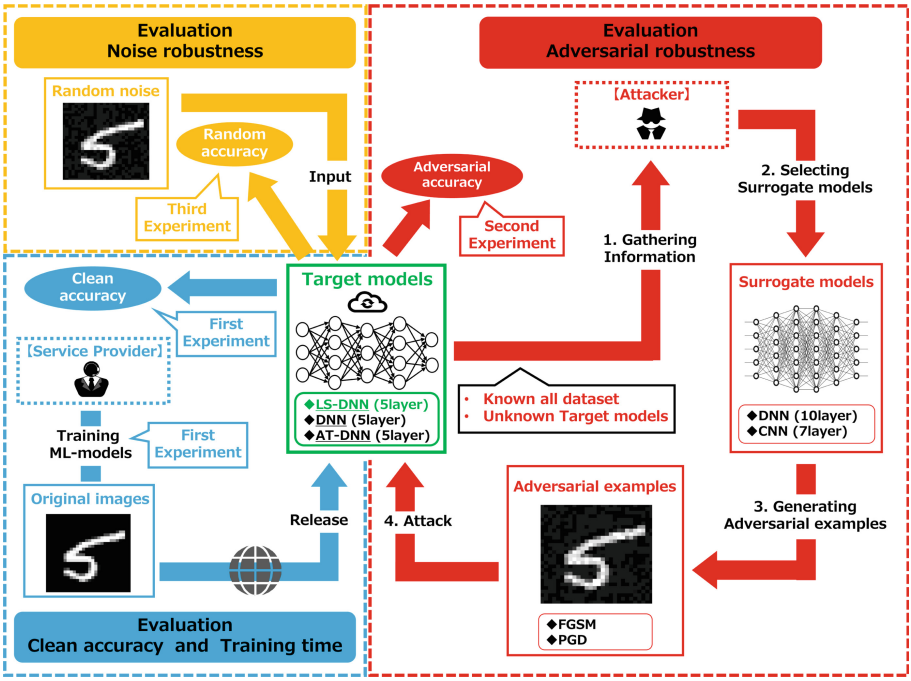


Fig. 3. Image of the Experiments (1st Experiment, 2nd Experiment, 3rd Experiment).

The purpose of the Exp-1 is to verify whether the reduction in Clean accuracy was suppressed and to measure the training time. The evaluation method of Clean accuracy is the accuracy of the Target models against 10,000 pieces of normal test data. The evaluation method of training time is the total CPU execution time from the start to the end of training on 60,000 training data. The purpose of the Exp-2 is to verify whether Adversarial robustness has been obtained. The evaluation method is the Adversarial accuracy of the Target models on 10,000 pieces of adversarial examples. We generate adversarial examples by FGSM and PGD using 10,000 normal all test data. Adversarial accuracy is then verified by continuously increasing the hyperparameter ε. The purpose of the Exp-3 is to verify whether Noise robustness has been obtained. The evaluation method is the Random accuracy of Target models against 10,000 pieces of random noise. We generate random noise by (9) using 10,000 normal all test data. Random accuracy is then verified by continuously increasing the hyperparameter ζ.

4.2 Conditions

Adversarial examples are generated by using framework Adversarial robustness Toolbox (ART) [13]. Target and Surrogate models architectures are shown in (Table 2).

Table 2. Surrogate and Target models architectures used in the experiments. Each layer indicates the type of connection and the magnitude of the output. For the CNN, the padding is set to 0, the stride is set to 1, the filter size to 3×3, and the number of filters are 32 and 64.

	Target models	Surrogate models	
Model Type	LS-DNN DNN AT-DNN	DNN	CNN
Layers	5	10	7
Input	Flatten(784)	Flatten(784)	28×28
hidden	Fully Connected(50)	Fully Connected(100)	Convolution (26×26×32)
	Fully Connected(50)	Fully Connected(100)	MaxPooling(13×13×32)
	Fully Connected(50)	Fully Connected(100)	Convolution (11×11×64)
	–	Fully Connected(100)	MaxPooling(5×5×64)
	–	Fully Connected(100)	Fully Connected(1600)
	–	Fully Connected(100)	–
	–	Fully Connected(100)	–
	–	Fully Connected(100)	–
Output	Fully Connected(10)	Fully Connected(10)	Fully Connected(10)
Activation	Sigmoid and Softmax	Sigmoid and Softmax	ReLU and Softmax

Target models architectures are three different models (LS-DNN, DNN, and AT-DNN), and their performance is compared. Surrogate models architectures are DNN and CNN. This is to verify whether the Adversarial robustness performance of the LS-DNN depends on Surrogate models architecture. We explain why we changed the number of hidden layers in the DNN of the Target and Surrogate models. The more similar Target and Surrogate models architectures are, the more susceptible they are to transferability [15] and the more difficult it is to verify accurate Adversarial robustness. For Target and Surrogate models, we set learning rate = 0.001, optimizer = RMSProp, epoch = 100, and minibatch = 50. Adversarial examples were generated continuously with ε ranging from $0.0 \sim 0.3$ in 0.05 increments. For PGD, $\alpha = 0.01$ and $n = 40$ in (6). AT-DNN is based on the previous study [1,12]. FGSM is set to $\varepsilon = 0.25$, PGD is set to $\varepsilon = 0.3$ and $\alpha = 0.01$. We trained different models with the ratio adjustment hyperparameter $\alpha = 0.1, 0.5, 0.9$ in (8). All other hyperparameters were set to the ART default values.

4.3 Results

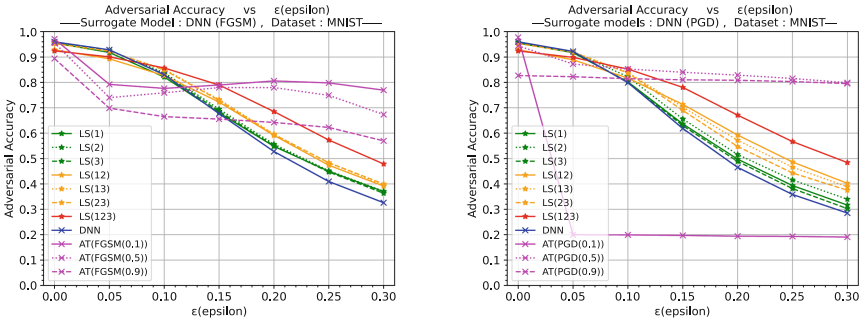
The results of Exp-1 are shown in (Table 3). Supplemental information about AT-DNN. We created six patterns of AT-DNN by using two different adversarial examples generation methods (FGSM and PGD) and three different $\alpha = 0.1, 0.5, 0.9$ in (8). For example, AT(FGSM(0.1)) is an AT-DNN that generates adversarial examples by FGSM and AT with $\alpha = 0.1$.

Table 3. Clean accuracy(Train(%) and Test(%)) and training time(Time(s)) against MNIST and Fashion-MNIST dataset of Target models: LS-DNN(Proposed method), DNN(Original), AT-DNN(Comparative method).

Target models		MNIST			Fashion-MNIST		
		Train(%)	Test(%)	Time(s)	Train(%)	Test(%)	Time(s)
LS-DNN (Proposed-method)	LS(1)	97.49	95.51	41,431	89.35	86.24	40,403
	LS(2)	98.02	95.74	41,518	89.83	86.74	41,362
	LS(3)	97.71	95.84	16,746	89.87	86.71	15,523
	LS(12)	93.66	92.87	71,493	87.84	85.31	72,408
	LS(13)	96.54	95.24	47,685	88.99	86.01	47,133
	LS(23)	96.41	95.28	47,745	88.33	85.59	49,988
	LS(123)	92.99	92.41	77,884	84.74	83.12	78,936
DNN (Original)	DNN	98.51	95.99	1,192	90.27	86.39	1,417
AT-DNN (Comparative-method)	AT(FGSM(0.1))	99.41	97.10	1,569	94.45	87.55	1,241
	AT(FGSM(0.5))	96.34	95.60	1,677	89.86	86.43	1,393
	AT(FGSM(0.9))	89.03	89.40	2,151	82.29	81.19	1,718
	AT(PGD(0.1))	99.17	97.73	11,031	92.24	87.72	12,610
	AT(PGD(0.5))	94.40	94.41	15,384	82.81	81.74	14,705
	AT(PGD(0.9))	82.83	82.70	28,407	65.73	65.44	27,935

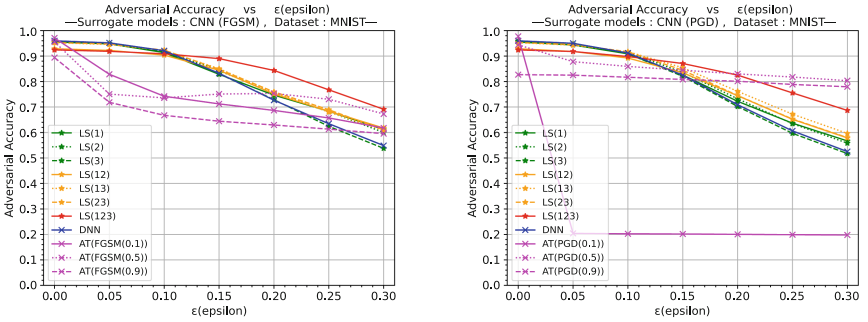
First, we will discuss Clean accuracy and training time in Exp-1. As for Clean accuracy, LS-DNN showed locally higher values than AT-DNN. In particular, for all LS-DNN, the values were higher than those of the AT(FGSM(0.9)) and AT(PGD(0.9)). However, LS-DNN tends to degrade when the number of LS Module is increased. This can be interpreted as overdone signal overwrite modification promoting degrade Clean accuracy. As for training time, compared to AT-DNN and DNN models, the training time increased significantly. The increase in computational complexity as the number of LS mechanisms increases can be attributed to the feedback function of the LS-DNN.

The results of Exp-2 are shown in (Fig. 4) $\sim$ (Fig. 7). It plots the change in the corresponding Adversarial accuracy when ε is increased by 0.05 from $0 \sim 0.3$ in FGSM and PGD. MNIST dataset shows in (Fig. 4) and (Fig. 5). Fashion-MNIST dataset shows in (Fig. 6) and (Fig. 7).



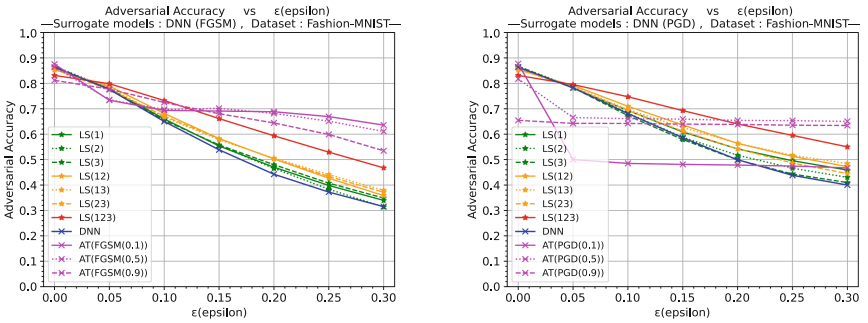
(a) FGSM Attack (b) PGD Attack

Fig. 4. Adversarial accuracy against MNIST (Surrogate models: DNN).



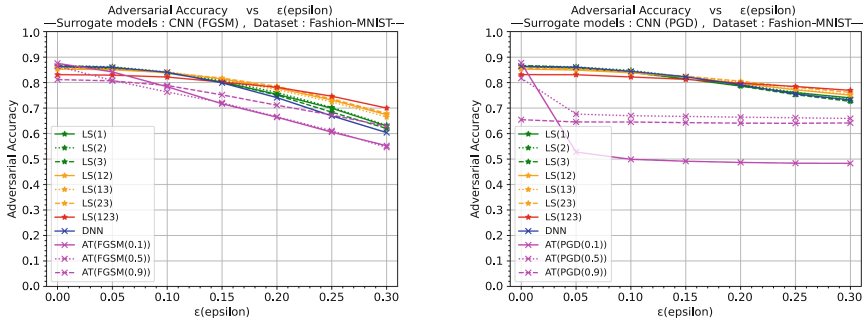
(a) FGSM Attack (b) PGD Attack

Fig. 5. Adversarial accuracy against MNIST (Surrogate models: CNN).



(a) FGSM Attack (b) PGD Attack

Fig. 6. Adversarial accuracy against Fashion-MNIST (Surrogate models: DNN).

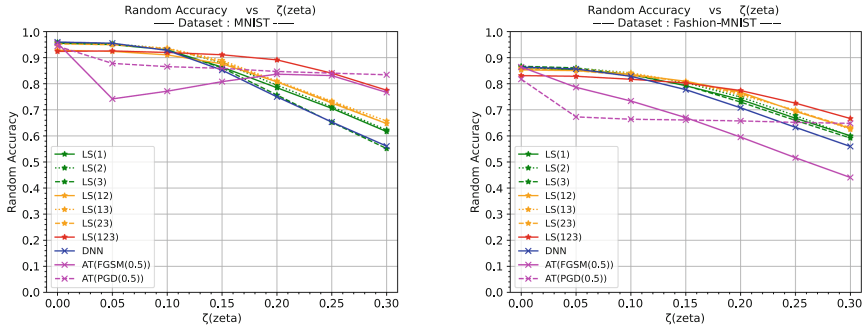


(a) FGSM Attack (b) PGD Attack

Fig. 7. Adversarial accuracy against Fashion-MNIST (Surrogate models: CNN).

Next, we will discuss two perspectives on Adversarial robustness in Exp-2. The first focus is on the number of times the LS Module is applied. We found that increasing the number of times the LS Module is applied tends to improve Adversarial robustness. Furthermore, this trend is independent of the dataset and Surrogate models architecture. It can be assumed that the more often an architecture processes signal overwriting, the more parameters are changed, and thus the gradient similarity with the Surrogate models can be relatively small [2]. The second focus is on the difference in Surrogate models architecture. When the Surrogate models is DNN, LS-DNN has weaker Adversarial robustness than AT-DNN, but stronger than DNN. On the other hand, when the Surrogate models is CNN, AT-DNN has weaker Adversarial robustness than both LS-DNN and DNN. These results were observed regardless of the type of dataset used. This suggests that LS-DNN have more generalized Adversarial robustness independent of Surrogate models architecture.

The results of Exp-3 are shown in (Fig. 8). This plots the change in the corresponding Random accuracy when ζ is increased by 0.05 from $0 \sim 0.3$. Note that the random noise is clipped between $[0, 1]$ to adjust the output.



(a) MNIST (b) Fashion-MNIST

Fig. 8. Random accuracy against MNIST and Fashion-MNIST.

Finally, we will discuss Noise robustness in Exp-3. Experimental results showed that LS-DNN tends to be less sensitive to random noise than DNN and AT-DNN. Moreover, we found that AT-DNN is sensitive to random noise not only in MNIST [23] but also Fashion-MNIST [8]. Those results may be due to overfitting for adversarial examples. The reason for the Noise robustness of human visual features is thought to be due to signal modification [10]. Therefore, the result that LS-DNN has Noise robustness suggests that LS-DNN may be able to acquire features similar to human visual representations.

5 Conclusion and Future Work

In this paper, we proposed to apply LS-DNN to Target models against Transfer-based attacks. The results showed that LS-DNN has Adversarial robustness independent of the Surrogate models architecture. We also showed that LS-DNN can overcome the problems of the AT, such as "degrade Clean accuracy" and "be overly sensitive to random noise."

Three issues for the future are listed below. The first is theoretical analysis. As Demontis et al. [2] point out, it is necessary to calculate the gradient alignment of the loss functions of LS-DNN and Surrogate models, and clarify their correlation. The second is the mitigation of "degrade Clean accuracy." We found that LS-DNN has a trade-off relationship between Adversarial robustness and Clean accuracy. Therefore, it is expected to optimize the number and placement of the LS Module. The third is reduction of the computation time for training. Compared to DNN and AT-DNN, LS-DNN is computationally more expensive due to its architecture. In parallel, we plan to investigate methods using the LS Module against attacks other than Transfer-based attacks.

References

1. Madry, A., Makelov, A., Schmidt, L., et al.: Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April - 3 May 2018, Conference Track Proceedings. OpenReview.net (2018)
2. Demontis, A., Melis, M., Pintor, M., et al.: Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In: 28th USENIX security symposium (USENIX security 19), pp. 321–338 (2019)
3. Chakraborty, A., Alam, M., Dey, V., et al.: A survey on adversarial attacks and defences. CAAI Trans. Intell. Technol. **6**(1), 25–45 (2021)
4. Xiao, C., Zhong, P., Zheng, C.: Enhancing adversarial defense by k-winners-take-all. In: 8th International Conference on Learning Representations. ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net (2020)
5. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
6. Tramèr, F., Papernot, N., Goodfellow, I.J., et al.: The space of transferable adversarial examples. CoRR abs/1704.03453 (2017)

7. Elsayed, F.G., Shankar, S., Cheung, B., et al.: Adversarial examples that fool both computer vision and time-limited humans. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
8. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747 (2017)
9. Taniguchi, H., Sato, H., Shirakawa, T.: Implementation of human cognitive bias on neural network and its application to breast cancer diagnosis. SICE J. Control Meas. Syst. Integr. **12**(2), 56–64 (2019)
10. Jang, H., McCormack, D., Tong, F.: Noise-robust recognition of objects by humans and deep neural networks. bioRxiv, pp. 2020–08 (2021)
11. Zhang, H., Yu, Y., Jiao, J., et al.: Theoretically principled trade-off between robustness and accuracy. In: International Conference on Machine Learning, pp. 7472–7482. PMLR (2019)
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y.Y.L. (ed.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
13. Nicolae, M.-I., Sinn, M., Tran, M.N., et al.: Adversarial robustness toolbox v0.2.2. CoRR abs/1807.01069 (2018)
14. Carlini, N., Wagner, D.A.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (S&P), pp. 39–57. IEEE (2017)
15. Papernot, N., McDaniel, P., Goodfellow, I.J.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. CoRR abs/1605.07277 (2016)
16. Papernot, N., McDaniel, P.D., Goodfellow, I., et al.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519 (2017)
17. Papernot, N., McDaniel, P.D., Jha, S., et al.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE (2016)
18. Russakovsky, O., Deng, J., Hao, S., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
19. Shinohara, S., Taguchi, R., Katsurada, K., et al.: A model of belief formation based on causality and application to n-armed bandit problem. Trans. Japan. Soc. Artif. Intell. **22**(1), 58–68 (2007)
20. Bhambri, S., Muku, S., Tulasi, A., et al.: A study of black box adversarial attacks in computer vision. CoRR abs/1912.01667 (2019)
21. Bai, T., Luo, J., Zhao, J., et al.: Recent advances in adversarial training for adversarial robustness. In: Zhou, Z. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19–27 August 2021, pp. 4312–4321. ijcai.org (2021)
22. LeCun, Y., Cortes, C., Burges, C.J.C.: MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Accessed 15 June 2022
23. Senzaki, Y., Ohata, S., Matsuura, K.: Negative side effect of adversarial training in deep learning and its mitigation. In: 2017 Information Processing Society of Japan, vol. 2017(2) (2017)