Unsupervised Adversarial Detection without Extra Model: Training Loss Should Change

Chien Cheng Chyou [1] **Hung-Ting Su** [1 2] **Winston H. Hsu** [1 3]

Abstract

Adversarial robustness poses a critical challenge in the deployment of deep learning models for real-world applications. Traditional approaches to adversarial training and supervised detection rely on prior knowledge of attack types and access to labeled training data, which is often impractical. Existing unsupervised adversarial detection methods identify whether the target model works properly, but they suffer from bad accuracies owing to the use of common cross-entropy training loss, which relies on unnecessary features and strengthens adversarial attacks. We propose new training losses to reduce useless features and the corresponding detection method without prior knowledge of adversarial attacks. The detection rate (true positive rate) against all given white-box attacks is above 93.9% except for attacks without limits (DF(∞)), while the false positive rate is barely 2.5%. The proposed method works well in all tested attack types and the false positive rates are even better than the methods good at certain types. The source code is available on ***this GitHub link***.

1. Introduction

Adversarial robustness, a model's ability to withstand adversarial attacks, is a critical issue for deploying deep learning models to real-world applications (Carlini et al., 2019). Adversarial training can strengthen adversarial robustness by using adversarial samples in training, but the side effects are still unbearable, such as significantly increased training time and the accuracy drop of the model (Goodfellow et al., 2014; Madry et al., 2017; Raghunathan et al., 2019; Zhang et al., 2019). What's more, adversarial training only works well for known attack types, which is not practical.

Another work line is adversarial detection (AD), which attempts to detect adversarial samples before the misled results are adopted. AD methods can be roughly separated into supervised and unsupervised methods (Aldahdooh et al., 2022). In supervised methods, it's possible to reach a satisfying accuracy for detecting adversarial samples. However, they can only detect the attacks they have already known, not to mention the extra training for classifying the given attacked samples (Aldahdooh et al., 2022). The state-of-the-art method now is LiBRe (2021), which reaches overwhelming detection accuracy and conquers most defects of other supervised methods. The critical problem is that what kind of attacks will come is unpredictable, and including all kinds of attacks is unrealistic.

We choose unsupervised AD as our research direction because of the unknown attack types in advance. However, current unsupervised methods are still very restrictive. Most unsupervised methods need extra models and rely on the distribution of features in the target model or even tune parameters differently based on the datasets or the attack types. Otherwise, they suffer from bad accuracy because the distribution of features changes with datasets and attack types (Aldahdooh et al., 2022). These methods are unaware of the redundant features learned from cross entropy. Therefore, we directly adjust how the target model learns the original classification task.

Different from the regular AD method, we make part of the target model's output unable to be attacked and detect adversarial samples by the attackable outputs, shown in 1. The key idea is that adversarial samples are a combination of the feature learned from the training dataset (Ilyas et al., 2019). The features for part of the model's outputs, which means these features can affect the corresponding outputs, are useless and can be removed, and then the corresponding outputs become unable to be affected by adversarial attacks. The model's raw outputs, the output before softmax, can be separated into two parts, the true outputs and the false outputs. The true outputs belong to the corresponding class to the input's class, and the false outputs do not. We find

[1]National Taiwan University [2]Columbia University [3]Mobile Drive Technology. Correspondence to: Chien Cheng Chyou <eternity6545@gmail.com>, Hung-Ting Su <htsu@cmlab.csie.ntu.edu.tw>, Winston H. Hsu <whsu@ntu.edu.tw>.

2nd AdvML Frontiers workshop at 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.
Copyright 2023 by the author(s).

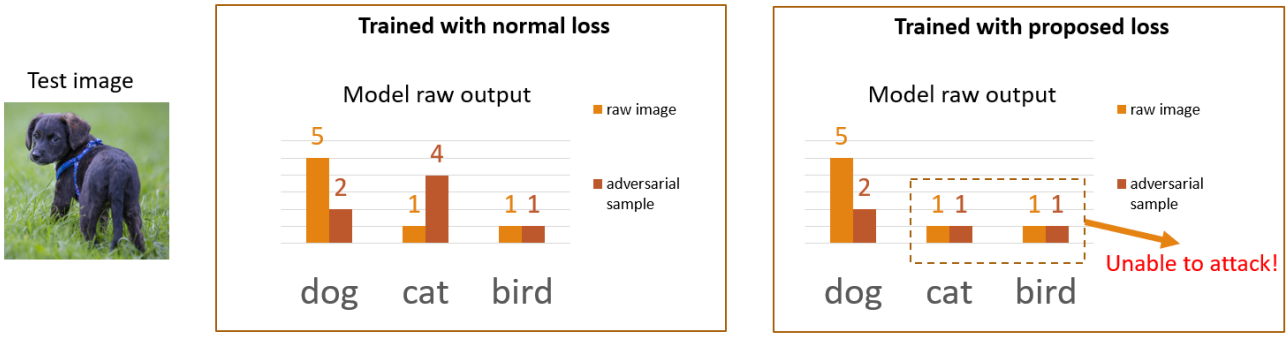

Figure 1. The effect of the proposed training loss and how to use it to detect adversarial samples. Because only the corresponding (dog in this case) raw output can be attacked and other outputs keep the same, the corresponding output becomes a signal to detect adversarial attacks. If no raw output is high, it means that the true output has been attacked.

that all model's raw outputs can be attacked with only L1 loss, like figure 2. That is, there are features for false outputs. Considering cross-entropy loss presses down false outputs, the feature for false output could be "why they are not something", but this is not a good way for humans to classify objects' categories. Human considers "why they are something", and add points to the corresponding class. Therefore, the feature for false outputs should be unnecessary and just give adversarial attacks more clues to attack the target model. This work wants to remove the features for false outputs by changing training loss, so false outputs become adversarially robust and only true outputs can be attacked. The true outputs' value is lower when attacks happen, so they become a signal for adversarial attacks.

We assume that directly keeping all the false raw outputs the same can remove the features for false outputs. The objective is achieved by forcing all the false raw outputs in a mini-batch to uniform distribution during training. The details will be explained later in section 3.2. The assumption works: the false raw output will be very hard to attack, and only the true output can be attacked. What's more, the range of false raw outputs is very separated from true raw outputs. Based on this property, we design a straightforward method for AD. If the maximum raw output is lower than the threshold or the minimum raw output is lower than the other threshold, the input should have been attacked already. There is no complicated algorithm, no extra model, and very low training overhead. The following are our contributions:

- We confirm that the behavior of cross entropy loss gives more features for adversarial attacks, and features for false outputs are unnecessary.
- We proposed a new training loss set that can effectively protect the false outputs from many kinds of white box adversarial attacks.

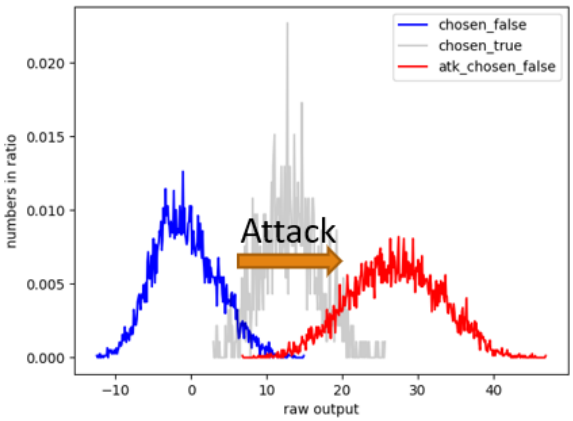

Figure 2. The histogram comparison of attacking class 0's false raw output, the output before softmax. This figure shows that even false raw output can be attacked easily and confirm that there are features for false outputs. Features for false outputs are unnecessary for humans and just give more clues for adversarial attacks. Therefore, this work wants to remove features for false output. PGD attack is applied to the false raw output of class 0 with negative L1 loss (pulling raw output up).

- We proposed a very simple but effective AD method based on the proposed training loss. Only the target model is used, and there is no extra model. The overhead of supervised and unsupervised detection methods no longer exists, such as extra training, extra computing, extra model weights, hyperparameter tuning, or trade-off for accuracy.
- In addition, the true positive rates are above 93.9% with a false positive rate of merely 2.5% against all given white-box attacks except for the attack without

constraint ϵ. All the attack types are unknown for the target model.

2. Related Work

Adversarial attacks aim to mislead the target model and force the target model gives the result they want. Depending on what model knowledge is known in advance, adversarial attacks can be categorized into three types, white box attacks, gray box attacks, and black box attacks (Aldahdooh et al., 2022). White box attacks mean everything about the target model is known, including the target model's structure, the training data, the weights, its inputs, and outputs. These attacks can directly compute gradient descent for some objective to get the input perturbations they want (Goodfellow et al., 2014; Kurakin et al., 2018; Moosavi-Dezfooli et al., 2016; Madry et al., 2017). Gray box (or semi-white box) attacks and black box attacks are not discussed in this paper. The strength of attacks is evaluated based on some hyperparameter: ϵ for the norm bound of attacks, how many iterations to update attacks, and how long a step goes.

AD counts on finding the special feature of adversarial samples. Supervised methods are a direct and effective direction. Statistics and patterns of the features can be found in adversarial samples, but they are not general to all kinds of attacks. On the other hand, the uncertainty methods seem to work better because they can deal with all kinds of attacks at the same time, just like LiBRe (2021). In this work, a lightweight bayesian neural network (BNN) is connected to a pretrain model, a classification model or even an object detection model, and refine the BNN model with given adversarial data from many types of adversarial attacks. This method solves the computing and training problem of BNN and reaches a spectacular accuracy for detecting the given kinds of adversarial attacks.

However, the type of adversarial attacks is unknown in the wild. That's why unsupervised methods should be applied in the real scenario. Unsupervised methods need to detect adversarial samples with only the original training data. Because the adversarial samples are unknown, the detection threshold is chosen based on the given false positive rate (FPR) for the clean testing data. There are too many different methods, so we only introduce the comparison methods in the experiments. Feature squeezing (FS) and MagNet are based on the property that if the input is attacked, the change in the input will change the output a lot (Xu et al., 2017; Meng & Chen, 2017). Deep neural rejection (DNR), Selective and feature based adversarial detection (SFAD) and Neural-network invariant checking (NIC) use the features in the target model to check whether the target model works properly (Sotgiu et al., 2020; Aldahdooh et al., 2023; Ma et al., 2019). In these unsupervised methods, the overhead is big. The hyperparameters are hard to tune and most need extra training and extra model, but their accuracies are still infeasible (Aldahdooh et al., 2022).

3. The Proposed Training Loss and Adversarial Detection

We find that there are features for false outputs shown in figure 2, and the goal is to reduce false output's features by constraining false raw outputs around the same value. In this way, only true outputs can be attacked and become the signals for adversarial attacks. True raw outputs will be high without attacks, and be low when attacks happen. The following will introduce the preliminary in section 3.1, and then explain how to reduce features for false outputs by the loss for false outputs in section 3.2. The training method is further improved by replacing the cross-entropy loss with the loss for true outputs in section 3.3 and removing the model's trend of one-hot output in section 3.4. After applying the proposed training loss, only the true output can be attacked. This property can be used to detect adversarial attacks and the method is introduced in section 3.5.

3.1. Preliminary

Kullback–Leibler divergence (KL divergence) will be the main subject of discussion. The objective of KL divergence is to measure the distance from a source probability distribution, usually the softmax outputs of a deep learning model, to a destination probability distribution. Cross entropy (CE) gets a different meaning in information theory, but CE's gradient is the same as KL divergence's. In most classification tasks, only one class is true, also called one-hot target.

When computing gradients of a deep learning model, the raw output vector $\boldsymbol{Y}$ from a sample after the last layer of the model is always followed with the softmax function σ to get the probability output vector $\boldsymbol{Y'}$:

$$\boldsymbol{Y'} = \boldsymbol{\sigma}(\boldsymbol{Y}) = \frac{e^{\boldsymbol{Y}}}{\sum_{y \in \boldsymbol{Y}} e^{y}} \tag{1}$$

Given the softmax output y' of a class and the label value l, the gradient of CE corresponding to the raw output y is very simple.

$$\frac{\partial CE}{\partial y} = y' - l \tag{2}$$

The gradient of raw output will be used to derive the proposed training loss in section 3.3.

3.2. Reduce Features for False Outputs

Constraining the false output around a certain value is hard because the suitable value is unknown. Applying KL divergence to learn a uniform distribution can constrain false

outputs to a suitable value automatically. We apply KL divergence to false outputs f from set F, which contains **all false outputs from all samples in a mini-batch**, and constrain them to uniform distribution U of false outputs.

$$Loss_{false_output} = -\sum_{f \in \boldsymbol{F}} U(f) \log \frac{\sigma_f(\boldsymbol{Y_F})}{U(f)} \tag{3}$$

After applying $Loss_{false_output}$, the range of false raw outputs becomes very narrow decided by the model itself automatically. The most important is that false raw outputs are hard to attack with negative L1 loss now.

3.3. Replace the Cross-entropy Loss

We find that pulling true outputs for each class through a mini-batch gets better results than the normal cross-entropy loss. The reason is that when some classes are less accurate, their true raw outputs prefer to be lower, and false raw outputs prefer to be higher. When applying CE, the softmax value of accurate classes and inaccurate classes are similar, so the gradients are also similar. Therefore, CE keeps training accurate classes while they are good enough, and inaccurate classes still fail to get higher gradients. The result is that true raw outputs from inaccurate classes are much lower. If true and false output values are mixed without attacks, we fail to identify the normal situation. The comparison is shown in section 4.2.1.

When pulling true outputs in each class, there are several true outputs and the number of them can be different in a mini-batch. The idea to get equivalent loss with CE is that given all false outputs are the same value, the true raw outputs with the same value should get the same gradient. Based on the equation 2, a raw output y should correspond to the same softmax output y' to get the same gradient $y' - l$.

At first, considering one true raw output with the value y_t and n number of false raw outputs with the same value y_f, we want to get the relation between y and y' for the true output. The softmax equation 1 can change to:

$$\begin{aligned} y'_t &= \frac{e^{y_t}}{\sum_{y \in \boldsymbol{Y}} e^{y}} = \frac{e^{y_t}}{e^{y_t} + \sum_{f \in \boldsymbol{F}} e^{y_f}} = \frac{e^{y_t}}{e^{y_t} + n \cdot e^{y_f}} \\ &= \frac{e^{y_t - y_f}}{e^{y_t - y_f} + n} \end{aligned} \tag{4}$$

Then, given a different false output's number n' in a different mini-batch, if we still want to get the same y'_t with the same y_t and y_f, we can just add a bias b to the true output:

$$\begin{aligned} \because y'_t &= \frac{e^{(y_t+b)-y_f}}{e^{(y_t+b)-y_f} + n'} = \frac{e^{(y_t+b-\log n'+\log n)-y_f}}{e^{(y_t+b-\log n'+\log n)-y_f} + n' \cdot \frac{n}{n'}} \\ &= \frac{e^{y_t - y_f}}{e^{y_t - y_f} + n} \\ \therefore b &= \log n' - \log n \end{aligned} \tag{5}$$

The relation between y and y' for the false output is changed but the gradient sum of false outputs is the same.

$$\sum_f \frac{\partial KL}{\partial y_f} = \sum_f y'_f = 1 - y'_t \tag{6}$$

In short, given one true output and n' false outputs, to obtain the equivalent loss to n false output, we just add a bias $b = \log n' - \log n$ to the true output.

On the other hand, the number of true outputs for a class in a mini-batch can change. The target now is to get the same relation between y and $y' - l$ when the number of true outputs is different. Considering that there are n_t true outputs with the same value y_t and n_f false outputs with the same value y_f, and we still get the relation between y and y'. The label of multi-hot objective is $1/n_t$ for true outputs:

$$\begin{aligned} y'_t - 1/n_t &= \frac{e^{y_t}}{\sum_{t \in T} e^{y_t} + \sum_{f \in F} e^{y_f}} - 1/n_t \\ &= \frac{e^{y_t}}{n_t \cdot e^{y_t} + n_f \cdot e^{y_f}} - 1/n_t \\ &= \frac{1}{n_t}\left(\frac{e^{y_t}}{e^{y_t} + \frac{n_f}{n_t} \cdot e^{y_f}} - 1\right) \end{aligned} \tag{7}$$

We can find the gradient is equivalent to one-hot objective averaged by the number of true outputs n_t, and $\frac{n_f}{n_t}$ is the averaged number of false outputs for each true output. That is, the gradient of multi-hot KL divergence with n_t true outputs and n_f false outputs equals to the gradient of the expected value of one-hot CE with 1 true output and $\frac{n_f}{n_t}$ false outputs.

In summary, we apply KL divergence to all outputs from all samples x_c of a class c, and average through each class with the number of classes N_C. The destination distribution is a multi-hot distribution M. We temporarily add biases vector $\boldsymbol{B_c}$, which depends on the false-true ratio of class c and is zero for false outputs, to the raw output vector $\boldsymbol{Y_c}$. That is:

$$Loss_{true_output} = -\frac{1}{N_C} \sum_c \sum_x M_c(x) \log \frac{\sigma_x(\boldsymbol{Y_c} + \boldsymbol{B_c})}{M_c(x)} \tag{8}$$

When using $Loss_{true_output}$, the false outputs from less accurate classes are compared with each other. They are similar to each other, so the punishment becomes lower.

Table 1. The true positive rate (TPR%) and false positive rate (FPR%) of different unsupervised AD methods against well-known white box attacks on CIFAR-10.

	FS		MAGNET		DNR		SFAD		NIC		**PROPOSED**	
ATTACK TYPE (ϵ)	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓
FGSM (8)	29.33	5.07	0.72	0.77	32.09	10.01	67.94	10.9	43.64	10.08	**100**	**2.5**
FGSM (16)	35.34	5.07	3.11	0.77	31.35	10.01	79.9	10.9	58.48	10.08	**100**	**2.5**
BIM (8)	8.74	5.07	0.56	0.77	4.27	10.01	18.12	10.9	**99.95**	10.08	**93.9**	**2.5**
BIM (16)	0.34	5.07	0.69	0.77	17.07	10.01	45.35	10.9	**100**	10.08	**95.8**	**2.5**
PGD (8)	8.2	5.07	0.57	0.77	11.34	10.01	29.49	10.9	**100**	10.08	**95.1**	**2.5**
PGD (16)	0.2	5.07	0.66	0.77	25.11	10.01	52.9	10.9	**100**	10.08	**99**	**2.5**
DF (∞)	39.18	5.07	57.33	0.77	30.2	10.01	89.57	10.9	**84.91**	**10.08**	78.8	2.5

Therefore, the raw outputs from less accurate classes are much higher than applying CE and help to get a lower false positive rate for the proposed AD method.

3.4. Remove the Trend of One-Hot Output

After applying the proposed two losses, we find that the target model prefers to keep one raw output high. When pressing down the maximum raw output with L1 loss by adversarial attacks, another raw output raises. On the other hand, pulling false raw outputs directly becomes much more difficult now.

To remove this trend, the idea is to generate the condition when every raw output should be low and penalize the output with a high value. To achieve this goal, we apply adversarial attacks to part of the training data in the mini-batch. The objective of adversarial attacks is only to press down the true raw output to generate the condition when one-hot output should not exist. For these adversarial samples, only the loss for false outputs is applied to penalize one-hot outputs. The ratio of adversarial samples can be small, 10% is enough. If the ratio is too high, the training time will become much longer and the accuracy of the original classification task will be lower.

3.5. The Proposed Adversarial Detection

After applying the proposed training loss, only the true output can be attacked. When adversarial attacks happen, the true raw output is pressed down with false outputs kept still. However, we do not know which output is the true output in the real scenario. Instead, we can only know the maximum raw output is much higher than other raw outputs in normal cases. When adversarial attacks happen, the maximum raw output will be much lower or even no longer the maximum one. In some cases, the original maximum raw output becomes the minimum one and is much lower than other raw outputs. Therefore, two thresholds are applied to check the maximum and the minimum raw output. An adversarial sample is detected when the maximum raw output is lower than the threshold or the minimum raw output is lower than the other threshold. The figure 3 shows why the proposed method works. Both thresholds are selected by the range of false raw output.

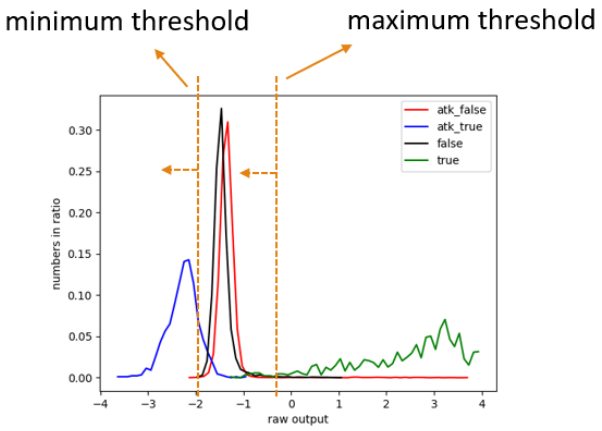

Figure 3. The histogram comparison of raw output before and after attacks. The target model is trained with the proposed loss. This figure shows that false outputs are very hard to attack (the "atk_" curve), so the range (the two thresholds) of false raw outputs can be used to test whether the value of true output is normal.

4. Experiments

The experimental details are in the appendix, including threshold choosing, training environment, training parameter, and training tricks.

4.1. Compared with Other Unsupervised Methods

Although adversarial attacks are used during training, the proposed method is not a supervised method. At first, the attacks during training are not learned directly. Only true raw

output is attacked in some data and only the false raw outputs of these data are included in the training loss. What's more, the applied attack type is unused in all regular attack types. The strength of this attack is also weaker than common attacks. Given $\epsilon = 8$ for infinite norm and 2 for an updated step length of PGD attack, the iteration is only 5 rather than 10 in most paper settings. The most important is that the proposed detection works well with every white-box attack we've tried, and supervised detection methods can't work with unknown attacks. Therefore, we consider the proposed method as an unsupervised method.

Following the results from the survey paper of aldahdooh (2022), the setting of adversarial attacks is a little different: the iteration of attack is 100, and there is no constraint ϵ of deep fool (DF) attack (Moosavi-Dezfooli et al., 2016). We choose infinite norm L_∞ for constraint ϵ. The test data of adversarial samples are generated by the Adversarial Robustness Toolbox (2018). The proposed method works extremely well against all types of the given white box attacks (Goodfellow et al., 2014; Kurakin et al., 2018; Moosavi-Dezfooli et al., 2016; Madry et al., 2017). Furthermore, the iterations are 100 rather than the common setting of 10, so the attack strength is much higher than the usual setting. The proposed method works worst in DF (∞) because the difference is too large (reaches 255 in some pixels) and even humans can see the difference. Under other attacks, the proposed method's true positive rate (TPR) defeats all other methods except NIC, but the false positive rate (FPR) is much lower than NIC. The result is in table 1.

4.2. Ablation Study

In the ablation study, AutoAttack (Croce & Hein, 2020) is also included because it can evaluate the ability of defense, also called adversarial robustness, more fairly than other attacks. The strength of attacks will follow the common setting:$\epsilon = 8$ with 10 iterations and $\epsilon = 16$ with 20 iterations. ϵ is based on an infinite norm, and the update rate is 2.

4.2.1. The Effect of Multi-Hot KL Divergence for Independent-Class Training vs Cross Entropy

The loss for true outputs can use the common CE loss, but it makes the results worse because some true raw outputs become lower. The table 2 shows that FPR is higher even with lower TPR.

Table 2. The true positive rate (TPR%) and false positive rate (FPR%) of the proposed method vs CE against well-known white box attacks on CIFAR-10.

	CE		THE PROPOSED LOSS	
ATTACK TYPE(ϵ)	TPR	FPR	TPR	FPR
PGD (8)	91.8	6.9	**94.2**	**2.5**
PGD (16)	85.1	6.9	**95.8**	**2.5**
DF (8)	99.5	6.9	**99.9**	**2.5**
DF (16)	93.5	6.9	**99.9**	**2.5**
AUTOATTACK (8)	79.2	6.9	**100**	**2.5**
AUTOATTACK (16)	79.4	6.9	**100**	**2.5**

4.2.2. The Effect of Removing the One-Hot Output Trend

When training without adversarial samples, the true raw outputs are higher in the classes with worse accuracy in the original classification task, and the accuracy of the original classification task is also higher. However, the trend of keeping one-hot outputs makes it vulnerable to any attack that presses down the true output, just like table 3. The TPR without adversarial samples is high against AutoAttack because the true outputs are lower than the minimum threshold.

Table 3. The true positive rate (TPR%) and false positive rate (FPR%) of the proposed method vs training without adversarial (adv.) samples against well-known white box attacks on CIFAR-10.

	WITHOUT ADV. DATA		WITH ADV. DATA	
ATTACK TYPE(ϵ)	TPR	FPR	TPR	FPR
PGD (8)	23.5	0	**94.2**	**2.5**
DF (8)	0.2	0	**99.9**	**2.5**
AUTOATTACK (8)	99	0	**100**	**2.5**

4.2.3. The Performance Comparison with and without the Minimum Output Threshold

The minimum output threshold is an engineering method to improve the accuracy of AD. The most noteworthy part is the ability of false raw outputs to defend against adversarial attacks, and it is still not perfect now. Here show the result without minimum output threshold and the corresponding AUROC to explain the ability of defense more briefly. Attackers may adjust true outputs carefully to avoid triggering the minimum output threshold, so improving the resistance ability of the false output is still the research key point.

Table 4. The true positive rate (TPR%), false positive rate (FPR%), and AUROC of the proposed method with and without the minimum output threshold against well-known white box attacks on CIFAR-10.

	WITHOUT			WITH	
ATTACK TYPE(ϵ)	AUROC	TPR	FPR	TPR	FPR
PGD (8)	97.8	92.1	2.5	**94.2**	2.5
PGD (16)	94.8	89.1	2.5	**95.8**	2.5
DF (8)	99.9	99.9	2.5	99.9	2.5
DF (16)	99.9	99.9	2.5	99.9	2.5
AUTOATTACK (8)	89.6	83.2	2.5	**100**	2.5
AUTOATTACK (16)	91.7	71	2.5	**100**	2.5

5. Conclusions and Future Works

We find that constraining false raw output to uniform distribution can reduce unnecessary feature learning for false outputs and help false output defend against adversarial attacks. Although keeping the model's result correct is still impossible, these adversarial samples can be easily detected by checking the maximum and the minimum raw outputs of the target model. Based on this property, We proposed an unsupervised AD method with high accuracy and much lower training time overhead than adversarial training. There is no extra model, extra training, extra computing, or difficult hyperparameter tuning. There is no accuracy trade-off for the original classification task.

What's more, the proposed training method may help in other ways. CE keeps learning accurate classes but neglects inaccurate classes, and the proposed loss for true outputs can solve this problem. Therefore, the proposed loss for true outputs may be useful when the dataset label is unbalanced or when some classes are harder to train.

On the other hand, we still do not understand why training with some adversarial samples will press down the true raw outputs. If this problem is solved, the FPR can be even lower and the detection method will finally be perfect.

Acknowledgements

This work was supported in part by the National Science and Technology Council, under Grant NSTC 110-2221-E-002-133-MY2, and Qualcomm through a Taiwan University Research Collaboration Project.

References

Aldahdooh, A., Hamidouche, W., Fezza, S. A., and Déforges, O. Adversarial example detection for dnn models: A review and experimental comparison. *Artificial Intelligence Review*, 55(6):4403–4462, 2022.

Aldahdooh, A., Hamidouche, W., and Déforges, O. Revisiting model's uncertainty and confidences for adversarial example detection. *Applied Intelligence*, 53(1):509–531, 2023.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Deng, Z., Yang, X., Xu, S., Su, H., and Zhu, J. Libre: A practical bayesian approach to adversarial detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 972–982, 2021.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pp. 99–112. Chapman and Hall/CRC, 2018.

Ma, S., Liu, Y., Tao, G., Lee, W.-C., and Zhang, X. Nic: Detecting adversarial samples with neural network invariant checking. In *26th Annual Network And Distributed System Security Symposium (NDSS 2019)*. Internet Soc, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Meng, D. and Chen, H. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 135–147, 2017.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., and Liang, P. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

Sotgiu, A., Demontis, A., Melis, M., Biggio, B., Fumera, G., Feng, X., and Roli, F. Deep neural rejection against adversarial examples. *EURASIP Journal on Information Security*, 2020:1–10, 2020.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

A. Experimental Details

When some samples' predictions are wrong, their maximum output without attack will be much lower even below the threshold. Therefore, to ensure the target model's accuracy not changing the accuracy of AD, we only adopt the test data with the correct answer before attacks.

Both thresholds for the proposed AD method are chosen based on false raw output distribution on CIFAR-10 test data. The maximum threshold is the 99th percentile, higher than 99% false raw outputs, and the minimum threshold is the lowest false raw output.

The training process runs on the RTX 3090 based on Tensorflow. The training dataset is CIFAR-10 (Krizhevsky et al., 2009). The model structure in the experiments is resnet18(He et al., 2016) customized for CIFAR-10. We translate the popular PyTorch code of resnet for CIFAR-10 into Tensorflow. The training epoch is 60. The training optimizer is SGD with a momentum of 0.9, and the learning rate is 0.1 for the first 50 epochs and 0.01 for the last 10 epochs. The proposed training loss keeps the accuracy of the original classification task almost the same, around 86%. The ratio of attack samples in the mini-batch is 10%. The attack samples used in training are generated by the PGD attack with $\epsilon = 8$ for infinite norm, 2 for an update step length, and 5 for update step iteration given the input value from 0 to 255.

Both the training losses are applied to raw outputs rather than softmax outputs. The training procedure needs to warm up with lower weight for false output's loss, 0.2 for the first 5 epochs and 1 for the last epochs. The weight for true output's loss keeps 1 during training. Notice that the kernel weight of the last layer of the model should be initialized with 0 to avoid overflow and the dead gradient of the loss for false outputs. In this work, the bias b is $\log \frac{n_f}{n_t} - \log 9$ because the number of false outputs in CIFAR-10 is 9. This setting helps compare the proposed loss for true outputs and CE. It also has to mention that the softmax in $Loss_{false_output}$ is very easy to overflow. When some values overflow, they are clipped to avoid infinite numbers or NaN values, and these outputs die because no gradient goes through them. A trick to deal with this is to initialize the kernel of the last layer with all zero or some similar ways. Besides, $Loss_{false_output}$ is fed with $\boldsymbol{Y_F}$ and $-\boldsymbol{Y_F}$ and average both to penalize the false raw outputs with extremely high or low values at the same time.