# Protect Your Deep Neural Networks from Piracy

Mingliang Chen
University of Maryland
College Park, USA
mchen126@terpmail.umd.edu

Min Wu
University of Maryland
College Park, USA
minwu@umd.edu

## Abstract

*Building an effective DNN model requires massive human-labeled training data, powerful computing hardware and researchers' skills and efforts. Successful DNN models are becoming important intellectual properties for the model owners and should be protected from unauthorized access and piracy. This paper proposes a novel framework to provide access control to the trained deep neural networks so that only authorized users can utilize them properly. The proposed framework is capable of keeping the DNNs functional to authorized access while dysfunctional to unauthorized access or illicit use. The proposed framework is evaluated on the MNIST, Fashion, and CIFAR10 datasets to demonstrate its effectiveness to protect the trained DNNs from unauthorized access. The security of the proposed framework is examined against the potential attacks from unauthorized users. The experimental results show that the trained DNN models under the proposed framework maintain high accuracy to authorized access while having a low accuracy to unauthorized users, and they are resistant to several types of attacks.*

## 1. Introduction

Deep Neural networks (DNNs) have gained a growing amount of attention, due to the significant performance leap in many learning problems ranging from computer vision to biomedical analytics [9, 6, 13, 2]. Building such a model that performs well is generally a substantial task, usually requiring massive human-labeled training data, powerful computing hardware and researchers' skills and efforts. Although some DNN models such as AlexNet [6], GoogLeNet [15], and VGGNet [14] are open to the public with permissions to use them for non-commercial purposes, many model owners in commercial applications expect to keep the trained DNN models private, due to the business considerations and/or the privacy and security issues. Therefore, the trained DNNs are becoming a new form of valuable intellectual property (IP) for the model owners. In order to encourage healthy business investment and competitions, the IP protection of these trained DNNs in commercial use is going to become increasingly important.

An important aspect of the IP protection is to ensure access control so that only authorized users can access the trained model and benefit from its high learning capabilities. This calls for the needs to investigate the techniques to enable access control to prevent unauthorized users from illicit use or embezzlement of the trained DNN models.

Recent research works have begun to address the security issues of DNN models. Inspired by the digital watermarking and fingerprinting techniques, recent studies in [16, 11, 12] introduced watermarks into DNN models to protect IP and claim the ownership without degrading the performance of the DNN models. The study in [10] proposed to identify the ownership of the DNN models by taking advantage of adversarial examples that lie near the decision boundaries of the models [3, 7]. The work in [18, 1] studied the impact of the poisoned training data on the performance of the trained DNN models. The authors in [18] embedded some designed watermarks into a portion of training data and assigned them a new label, leaving backdoors in the models. The authors in [1] discussed multiple approaches to generate the poisoned training data and evaluate their influence on the trained DNN models.

Although the above methods, to some extent, can facilitate the identification of ownership of the DNN models during a post-piracy investigation, they do not actively address the problem of unauthorized access and piracy/embezzlement. For example, an intruder or an insider may steal the whole DNN model and make a fortune from it, which cannot be easily prevented by conventional password-based access control to the computing system. Hence, it is important to investigate mechanisms to protect the DNN model itself against unauthorized copy and use. In addition to piracy protection, a proper access control also supports privacy protection against the misuse of DNNs that produce decision results of a private or sensitive nature.

One intuitive way is to encrypt the weights of the DNN

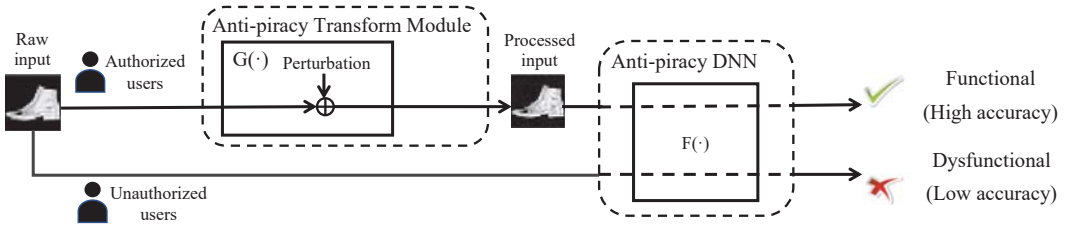



Figure 1. The proposed framework

model with traditional data encryption methods such as RSA or advanced encryption standard (AES). However, this would mean that for the DNN model to be run properly, either the DNN model needs to be put into a trusted computing platform using trusted platform modules (TPMs) so that all the encrypted model parameters can be decrypted and then securely run to provide results; or the DNN computations directly on the encrypted parameters must be enabled through homomorphic encryption or other types of encrypted-domain computation tools. Given the large number of parameters, on the order of millions or more, involved in many state-of-the-art DNNs and the overall high computational power required, both of these security strategies would be very expensive under today's secure computing paradigm. It is challenging to fully utilize the off-the-shelf optimized DNN architectures and hardware-software tools designed for learning in plaintext while simultaneously achieving security and efficiency.

In this paper, we investigate how to tackle this challenge by dividing a DNN system into two parts. More specifically, we maintain the bulk part of the DNN that can be run in a similar way as today's ordinary DNN utilizing the available software-hardware platform and only produce state-of-the-art result when taking in an "authorized" type of input, and in the mean time, we design a transformation module that can provide such "authorized" inputs at a reasonable level of computational complexity. By dividing the overall system into these two parts, we can concentrate the security resource to protect the transformation module, for example, through a trusted computing platform.

The next question is what kind of the transformation can be applied in the module. We are inspired by the adversarial examples whereby well-designed noise in a small strength can render a poor performance of a DNN model [3, 7]. If the function of the transformation module is adding adversarial perturbation to the inputs, would it be possible to design a DNN so that when an unauthorized user presents an input to the trained model, it would be as if it is contaminated by adversarial noise and thus lead to a poor outcome? And in contrast, an authorized user can "pre-process" the input using the transformation module, so that the performance approaches the state-of-the-art.

To this end, we propose a new framework and investigate anti-piracy algorithms to provide a transformation which is applied to the input and develop the matching DNN model, aiming at making DNN models behave properly only with the model owners or those authorized by the owners. In the proposed framework, we take advantage of adversarial examples and use them to provide access control to the trained DNN models. To the best of our knowledge, this is the first attempt to provide a framework and effective algorithms to design DNN models for access control purpose using adversarial examples.

## 2. Proposed Framework

In order to mitigate the impact of piracy, we propose a framework, presented in Figure 1, to train a DNN model with special characteristics. Specifically, the DNN model is functional with high recognition accuracy for authorized access, while it is dysfunctional for unauthorized users.

We introduce an *anti-piracy transform module* into our proposed framework, whose function is to embed specific verification of the authorized access, waiting for the following anti-piracy DNN to recognize. We shall refer to the input as the *raw input* and the output of the anti-piracy transform module as the *processed input*. Note that, it may be possible to make the anti-piracy DNN architecture known to the public, while only authorized users are provided access to the anti-piracy transform module so that their inputs will go through such a process beforehand.

As shown in the block diagram in Figure 1, the classification result has a different accuracy rate depending on whether the access is from an unauthorized user or an authorized one. Our proposed anti-piracy framework has two components: the anti-piracy transform module and the anti-piracy DNN which can recognize the authorized and unauthorized inputs and provide differential performance, respectively, thus enabling access control. Conceptually, the anti-piracy transform module mimics a "decryption" module, allowing the authorized users to generate the appropriate input for the DNN to function properly. As stated in the introduction, the anti-piracy transform module can be secured via TPM. With such a focused protection on the anti-piracy transform built into our proposed framework, optimization and customization can be done in accordance to specific applications' needs.

### 2.1. Threat modeling

We consider several types of adversaries in the design of the anti-piracy transform module. A simple, *opportunistic* attack is to directly copy and steal the anti-piracy DNN model. This type of attack is common and feasible by even the least resourceful adversaries when no protection is applied to the trained DNN model.

A more advanced adversary may attack the proposed anti-piracy transform module by generating the perturbation patterns to emulate the effect of the transformation, aiming to fool the anti-piracy DNN as if the inputs are from authorized users. Such attacks can be categorized into two scenarios: *Input-only attack* and *Pair attack*. In the "Input-only attack" scenario, the adversary can only access the raw inputs but can hardly wiretap the outputs of the anti-piracy transform module. For instance, the connection between the anti-piracy transform module and anti-piracy DNN is secured with encryption or encapsulated within a secure tamper-proof hardware packaging. As for the "Pair attack" condition, the adversary is assumed to successfully obtain the input-output pairs of anti-piracy transform module.

### 2.2. Formulation to construct anti-piracy network

To facilitate the discussions, we first define some notations. Let $x_r$ be the raw input and $x_p$ the processed input. $G(\cdot)$ denotes the anti-piracy transform module, thus $x_p = G(x_r)$. $F(\cdot)$ denotes the anti-piracy DNN. As we have mentioned, the anti-piracy DNN gives a learning outcome at a low accuracy with the raw input $F(x_r)$, while it gives a higher accuracy rate with the corresponding processed input $F(x_p)$.

To ensure the anti-piracy DNN functional to the processed input $x_p$, we define the loss for $x_p$ as the conventional cross-entropy loss:

$$E_p = -\sum_{i=1}^{N} p_i \log q_{p,i} \tag{1}$$

where $N$ is the number of class, the vector $(p_1, p_2, ...p_N)$ is the one-hot encoding ground truth, and the vector $(q_{p,1}, q_{p,2}, ...q_{p,N})$ is the softmax output of $F(x_p)$.

To achieve the goal of the model being dysfunctional to the raw input $x_r$, we define the loss for $x_r$ as

$$E_r = \sum_{i=1}^{N} p_i q_{r,i} \tag{2}$$

where the vector $(q_{r,1}, q_{r,2}, ...q_{r,N})$ is the softmax output of $F(x_r)$. Since the vector $(p_1, p_2, ...p_N)$ is the one-hot encoding of the ground truth in terms of popular machine learning implementations, Equation (2) indicates the probability that the result is correct for the raw input $x_r$. If $E_r$ is minimized, the result is in low accuracy.

Combining the above two parts, we formulate the loss function $E$ as

$$E = \alpha E_p + \beta E_r + \gamma \left\| x_p - x_r \right\|_2^2 \tag{3}$$

where $\alpha$ and $\beta$ are two hyperparameters to balance the two loss terms with respect to the raw input and the corresponding processed input, and $\gamma$ is the regularization coefficient.

The regularizer in the third term is utilized to confine the generated perturbations in a small range. The reason for such a restriction is to preserve the processed input set similar to the raw input set as much as possible. Otherwise, the classification task for our framework can be substantially different from that for which the corresponding DNN structure is initially designed. Hence, the regularizer can help retain the comparable learning performance between our framework and the corresponding DNN structure. By minimizing the loss function $E$ in (3), we can obtain the DNN model which satisfies the goals proposed in Section 2.

### 2.3. Anti-piracy transform

The anti-piracy transform is a key component in our framework. Inspired by adversarial examples for DNN as motivated in the introduction, we design this module via applying perturbations on the raw input. The processed input will be the raw input incorporating the perturbation pattern that we design. We progress our investigation in this paper by considering three versions of anti-piracy transforms, from simpler to more sophisticated ones.

The first version is the *Fixed* approach, which generates a universal perturbation matrix beforehand by the owners. Specifically in this paper, we use a bipolar perturbation in the Fixed approach, whereby the amplitude of the perturbation in each pixel is taken from $\{\sigma, 0, -\sigma\}$ with probability $\{p, 1-2p, p\}$, respectively.

One possible drawback is that a specific instance of perturbation might not be an "optimal" perturbation for the whole framework, which motivates us to propose the *Learned* approach. Just as its name implies, the Learned approach aims at finding the optimal universal perturbation matrix for all input instances and achieving the best differential in the learning performance between authorized vs. unauthorized users.

However, the perturbation matrices in these two versions are easy to crack even when one input-output pair of the anti-piracy transform module is available to an adversary, since the perturbation is identical to all raw inputs so that the subtraction between the pair is enough to recover the perturbation. To mitigate this disadvantage, the *Generator* approach is formulated, which is an input-dependent perturbation generator. The Generator approach can take any form, such as a fully-connected network, a convolutional network, or other parameterized forms, and generate a specific perturbation matrix for each input.

Table 1. The structure of simple CNN used in the paper.

| Layer | Output size | Building block |
|---|---|---|
| conv1 | $28 \times 28$ | $[3 \times 3, 32]$ |
| pool1 | $14 \times 14$ | max, $2 \times 2$ |
| conv2 | $14 \times 14$ | $[3 \times 3, 64]$ |
| pool2 | $7 \times 7$ | max, $2 \times 2$ |
| fc1 | 1024 | dropout: 0.5 |
| fc2/output | 10 | softmax |

In the following sections, we will examine the classification performances of the three anti-piracy transform modules and their vulnerability against adversaries.

## 3. Experiments and Results

In this section, we conduct experiments to demonstrate the effectiveness of our proposed framework. In the proof-of-concept experiments, we used three well-known datasets of the MNIST [8], the Fashion [17], and the CIFAR10 [5].

### 3.1. Network structures and training settings

We used simple convolutional neural network (CNN) and Resnet-20 [4] as the architecture of the anti-piracy DNN. The detailed structure of simple CNN is shown in Table 1 and the structure of Resnet-20 is described in [4]. If not specified otherwise, the rectified linear unit (ReLU) is used as the activation function in each layer.

Since all the components in our proposed framework is differentiable, we used stochastic gradient descent (SGD) optimizer to train the anti-piracy DNN under the proposed framework. We set the hyper-parameters $\alpha = 1$, $\beta = 1$ and $\gamma = 0.01$. The initial learning rate was set to 0.1 and divided by 10 every 10k iterations, the weight decay to $5 \times 10^4$, the momentum to 0.9 and the batch size to 128. The models were trained up to a total of 40k iterations.

### 3.2. Experimental results

We trained the anti-piracy DNN on three datasets with multiple DNN architectures in Section 3.1 using three anti-piracy transform modules in Section 2.3. To show that our proposed framework does not degrade the performance of the DNN structures, we also trained the DNN without the anti-piracy design for a baseline reference. For the MNIST and the Fashion datasets, we normalized the grayscale images to the range $[0, 1]$. For CIFAR10 dataset, we normalized the color images to the range $[-1, 1]$ and follow a simple data augmentation: a $28 \times 28$ crop was randomly sampled from the original image or its horizontal flip. We only evaluated the single view of the $28 \times 28$ middle crop of the images in CIFAR10. Table 2 presents the performances of the proposed framework on different datasets. For the three proposed anti-piracy transform modules, the values outside the parentheses are the performance to the authorized access (i.e., feeding the transformed input into the DNN), and the values inside the parentheses are the prediction accuracy to the unauthorized access (i.e., feeding the raw input).

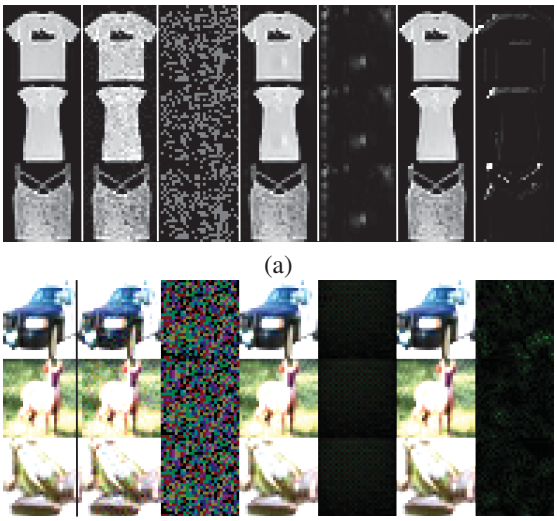
Table 2. The performance of the proposed framework.
The numbers in the parentheses are the performance by unauthorized access

| | Dataset | | | |
|---|---|---|---|---|
| | MNIST | Fashion | Fashion | CIFAR10 |
| Model | simple CNN | | Resnet-20 | |
| Baseline | 99.12% | 91.80% | 92.63% | 90.74% |
| Fixed | 99.24% (0.24% ) | 91.88% (1.09%) | 91.65% (0.63%) | 89.73% (0.52%) |
| Learned | 99.18% (0.10%) | 92.06% (2.18%) | 92.56% (0.65%) | 90.58% (0.86%) |
| Generator | 99.23% (0.23%) | 91.82% (2.76%) | 92.55% (1.55%) | 90.61% (0.78%) |

In the Fixed approach, we set the amplitude $\sigma = 0.1$ and probability $p = 0.2$. In the Generator approach, we form the perturbation generator with the structure of a convolutional layer with a filter size of 5-by-5, cascaded by a bottleneck layer with a filter size of 1-by-1. The number of channels in the intermediate layer is 16 for the MNIST and the Fashion datasets, and 64 for the CIFAR10 dataset. We used *tanh* function as the activation in the output layer.

From Table 2, we can see that three proposed perturbation-based anti-piracy transformations satisfy all the anti-piracy requirements. The anti-piracy DNN models under authorized access have comparable performances to the corresponding baseline DNN models. Actually, some performances of the anti-piracy DNN models even exceed the baselines. On the other side, the values in the parentheses indicate the anti-piracy DNN models are dysfunctional with the raw inputs.

For the Fixed method, under authorized access, we obtain the testing accuracy of $99.24\%$, $91.88\%$, $91.65\%$ and $90.84\%$, respectively, in four anti-piracy DNN models. We can see small drops of $1\%$ in accuracy in the third and fourth models compared with the corresponding baselines. A possible reason of this small drop is that these two models might not have obtained a good candidate of the universal perturbation pattern, since the Fixed method randomly generates the pattern beforehand, suggesting the benefit of searching for a good perturbation pattern. In practical situations, as the complexity of datasets and models increases, it would become increasingly difficult to generate a satisfactory universal perturbation. To overcome this problem, the other two approaches that we have considered can learn the perturbation pattern during the DNN training process. Because of the learning strategy, the models can obtain the optimal perturbation patterns and the anti-piracy DNNs. The



(a)

(b)

Figure 2. Examples of raw inputs and processed inputs fed into the anti-piracy DNNs. Column 1 is the raw inputs. Column 2 and 3 are the processed inputs and the perturbation in the Fixed method. Column 4 and 5 are the processed inputs and the perturbation in the Learned method. Column 6 and 7 are the processed inputs and the Generator method. (a) Simple CNN model on the Fashion dataset. (b) Resnet-20 model on the CIFAR10 dataset.

experimental results reflect that the gaps are small between the performances of the anti-piracy DNN models and the corresponding baselines in the Learned and Generator approaches, suggesting that the anti-piracy consideration in our framework impose negligible penalty in the the learning performance for authorized users.

### 3.3. The raw inputs and the processed inputs

In the proposed framework, we obtain the anti-piracy DNN model that can differentiate the raw inputs and the processed inputs and classify them to achieve a significantly different accuracy. In this subsection, we examine the relationship between the raw inputs and the processed inputs. Figure 2 shows the raw inputs and the processed inputs on different anti-piracy models with different anti-piracy transform methods. To facilitate the examination of the difference between raw inputs and processed inputs, the amplitude of the difference is magnified by 5 times. As indicated in Figure 2, the raw input and the corresponding processed input appear to be very similar, and human eyes can hardly distinguish them. This observation suggests the capability of the anti-piracy DNN models to differentiate the raw inputs and the processed inputs even though the perturbation is nearly imperceptible, similar to the effect of many adversarial examples demonstrated in the recent literature. Our proposed framework takes advantage of the sensitivity of the DNN models to protect the DNN models from piracy, and demonstrates its effectiveness to prevent the unauthorized access.

Figure 2 also reveals differences among the three perturbation-based anti-piracy transform modules. The last two methods are capable of generating much subtler perturbation patterns, thanks to the learning strategy applied in the module. In addition, the Generator method can provide the optimal individual perturbation pattern for inputs with the help of the training samples, which increases the complexity of the anti-piracy transform module.

## 4. Discussions

**Performance against piracy attacks** The performance against piracy attacks is an important aspect to examine for our proposed framework. We have investigated the performance against piracy attacks in the context of the Resnet-20 models trained on the Fashion dataset and present the result in Table 3. As discussed earlier, "Direct piracy" refers to directly copy the anti-piracy DNN model; "Input-only attack" refers to attacks only with the raw inputs; and "Pair attack" refers to attacks with the raw and corresponding processed input pairs. For each attack method, we have conducted experiments 100 times and recorded both the average and the best attack performance.

Under the direct piracy, the accuracy of the models is around $1\%$, indicating that direct piracy of the DNN model merely leads to invalid classification results. This suggests that the proposed framework is immune to this adversary.

In the "Input-only attack" scenario, an adversary is expected to generate a universal perturbation to mimic the function of the anti-piracy transform modules. Here, we assume that the adversary knows the settings of the universal perturbation generation and applies the same strategy, i.e., a bipolar perturbation with the same parameters $\sigma$ and $p$ stated in Section 3.2.

The results in the column of "Input-only attack" in Table 3 show that these Input-only attacks increase the classification accuracy of the anti-piracy DNNs with the Fixed and Learned modules. We believe that this is because the "Input-only attack" strategy mimics the first two methods to apply universal perturbations to all raw inputs, and eventually attenuates the differentiation ability between the processed inputs and the fake processed inputs. According to the best attack performances, these two models have a comparable vulnerability to "Input-only attack" strategy, nevertheless, there are still more than $10\%$ performance gaps between the models after "Input-only attack" and the models in the anti-piracy framework. On the other side, even after the attack, the DNN with the Generator module still provide a lower classification accuracy by a considerable amount for unauthorized users than an authorized user can obtain, indicating that the Generator approach can defend such an attack.

Table 3. The performance of the anti-piracy DNN models under various attacks.

| Model (Accuracy) | Direct piracy | Input-only attack | | Pair attack | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 10% | | 50% | | 100% | |
| | | Mean | Best | Mean | Best | Mean | Best | Mean | Best |
| Fixed (91.65%) | 0.63% | 66.23% | 78.96% | - | - | - | - | - | - |
| Learned (92.56%) | 0.65% | 55.37% | 79.42% | - | - | - | - | - | - |
| Generator (92.55%) | 1.55% | 3.17% | 4.95% | 75.05% | 82.11% | 76.31% | 84.17% | 77.24% | 86.00% |

As for "Pair attack", the models with the Fixed and Learned approaches are expected to be vulnerable and circumvented with only one input-output pair of the anti-piracy transform module, since the perturbation in each input is identical in these two methods. However, the Generator method avoids such one-pair-crack predicament. We assume that the adversary knows the structure of the anti-piracy transform module in the Generator method, but does not know the parameters. The task for the adversary is to recover the anti-piracy transform module with the help of a number of input-output pairs. The adversary can learn the parameters in the transform module with the given pairs as the input and the ground truth.

We have examined the model with the Generator method under the "Pair attack" using $10\%$, $50\%$ and $100\%$ of the pairs from the training set, respectively. As shown in Table 3, the "Pair attack" manages to cheat the anti-piracy DNN to some extent, increasing the accuracy from $1.55\%$ to around $85\%$ in the best case from an adversary point of view, compared to "Direct piracy". This means that the "Pair attack" strategy is capable of confusing the anti-piracy DNN model to some extent with the processed inputs and the fake processed inputs. However, this type of attack still fails to achieve the comparable accuracy with the model in the anti-piracy framework, which is $92.55\%$ in our example. Furthermore, the number of pairs used in the "Pair attack" influences the attack performance positively. Specifically, when the number of pairs increases from $10\%$ to $100\%$ of the training set, the value enhances slightly from $75.05\%$ to $77.24\%$ in average performance and from $82.11\%$ to $86.00\%$ in the best case scenarios for the adversaries.

To provide a context to the classification accuracy, in today's deep model R&D, a $1\%$ boost in the final performance is often considered as a significant improvement or even a breakthrough. Hence, the $5-15\%$ performance gaps between the model in the anti-piracy framework and those after various adversarial attacks demonstrate the resistance of the proposed framework to the various attacks.

**Relation between the baseline DNN and the anti-piracy DNN** The conventional training of the DNN model often requires the DNN model to learn the features of the raw input. When we fix the pre-trained DNN model and add well-designed small perturbations to the raw inputs, these adversarial examples are able to fool the model by rendering the classification outcome deviated significantly from the result before applying the perturbations. The baseline DNN models are thus valid to the raw inputs but become ineffective to their adversarial examples.

In our proposed framework, we swap the roles of the raw input and adversarial examples in the baseline DNN models. The anti-piracy DNN model learns the appropriate features of the perturbed raw inputs (i.e., the processed inputs), and the raw inputs effectively become the adversarial examples to the anti-piracy DNN model. Hence, the problem we need to solve here is given the adversarial examples, to find the equivalent "raw inputs" and their corresponding DNN model. Our proposed framework provides a feasible transformation module to produce such an equivalent "raw input" (i.e. the anti-piracy transform module) and the trained DNN (i.e., the anti-piracy DNN) that can classify the equivalent "raw input" in high precision and the "adversarial examples" in low accuracy.

**Areas of improvement** Conceptually, the proposed work shares some connections with data obfuscation. Investigating the role of established cryptographic building blocks into our proposed framework may help provide potential improvements, for example, to provide more explicit use of a cryptographic key. In our current implementations, the parameters in the anti-piracy transform modules may play some role of a key. Introducing the "key" concept explicitly into the anti-piracy framework will help improve the security strengths and facilitate security analysis.

## 5. Conclusion

In this paper, we have introduced a framework of utilizing the adversarial behavior of the DNN and turning it into a protection tool against misuse and piracy. The framework consists of an anti-piracy transform module and the corresponding anti-piracy DNN. The anti-piracy DNN is only valid to the authorized users who are provided the proper transformation. Any illicit use of the models will lead to DNN results with a low classification accuracy. The experimental results show the effectiveness of the proposed framework to protect the trained DNN models from piracy, and the anti-piracy DNN models are resistant to several types of attacks. Additional research along this line in the future can help develop a systematic understanding and refine the protection strategies.

## References


[1] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 1

[2] 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*, 542(7639):115, 2017. 1

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4

[5] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 4

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pages 1097–1105, 2012. 1

[7] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1, 2

[8] Y. LeCun. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998. 4

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015. 1

[10] E. L. Merrer, P. Perez, and G. Trédan. Adversarial frontier stitching for remote neural network watermarking. *arXiv preprint arXiv:1711.01894*, 2017. 1

[11] Y. Nagai, Y. Uchida, S. Sakazawa, and S. Satoh. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval*, 7(1):3–16, 2018. 1

[12] B. D. Rouhani, H. Chen, and F. Koushanfar. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750*, 2018. 1

[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 1

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1

[16] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277. ACM, 2017. 1

[17] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 4

[18] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172. ACM, 2018. 1