Adv-watermark: A Novel Watermark Perturbation for Adversarial Examples

Xiaojun Jia
jiaxiaojun@iie.ac.cn
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China

Xiaochun Cao∗
ciaoxiaochun@iie.ac.cn
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China

Xingxing Wei∗
xxwei@buaa.edu.cn
Beijing Key Laboratory of Digital Media, School of Computer Science and Engineering, Beihang University
Beijing, China

Xiaoguang Han
hanxiaoguang@cuhk.edu.cn
Shenzhen Research Institute of Big Data, the Chinese University of Hong Kong (Shenzhen)
Shenzhen, Guangdong, China

KEYWORDS
Adversarial Examples, Watermark Perturbation, Basin Hopping Evolution

1 INTRODUCTION
Recent literature has found that Deep Neural Networks (DNNs) are vulnerable to the adversarial examples which are generated by adding some imperceptible noises to the clean images [5]. Generally speaking, attack methods can be divided into two categories: white-box attack methods and black-box attack methods. The white-box attack [2, 6, 15, 23] denotes that the attacker has complete access to the target model such as model parameters, model structure, etc. And the black-box attack [1, 4, 17, 29, 33] denotes that the attacker can only access the output of the target model. The above methods achieve attacks by generating imperceptible perturbations. They use $L_0, L_2, L_{\infty}$ to bound the noises. Recently, more and more researchers pay attention to generating realistic adversarial examples without the $L_p$ norm limitation [4, 16, 26].

Watermarking methods [7] play an important role in protecting intellectual property rights. It embeds some specific information of the copyright holder (such as university logos, ownership descriptions, etc) into the multimedia data according to the requirements of users. In [21], Mintzer et al. describe the characteristics of visible watermarks. The visible watermark should be visible but does not significantly obscure the details of the host image.

In this paper, we propose a novel adversarial attack which generates adversarial examples using watermarks. We find that although watermarks do not affect people’s understanding of the image content, and adding specific watermarks to the clean images can fool the DNN models. The specific watermarks refer to the specific position and transparency of them. We mainly consider using visible watermarks to generate adversarial examples. In detail, we use alpha blending [31] to achieve watermark embedding. The host...
image and the watermark are multiplied by a scaling factor. The scaling factor is manipulated in the $\alpha$ channel of the image, which decides the image’s transparency.

As for a certain watermark, the DNN models can be successfully attacked only by adding the watermark with the specific transparency to a specific position of the host image. Considering this, we propose a novel attack method to generate watermark adversarial perturbations. Specifically, we propose a Basin Hopping Evolution (BHE) algorithm to find the appropriate transparency of the watermark image and the appropriate position within the host image to embed watermark. BHE is proposed based on the Hopping Evolution (BH) [36], where we find it usually falls into a local optimum and fails in attacking DNN models. In contrast, BHE has multiple initial starting points and crossover operation to keep the diversity of solutions. In this way, BHE makes it easier to find a global optimal solution and thus achieves a higher attack success rate than BH. The proposed method achieves attacks by using a little information (predicted probability of the classification model). It does not need the inner information of DNNs such as network structures and weights. Therefore, it belongs to the black-box attack.

Besides the ability to perform adversarial attacks, Adv-watermark also inherits the function of the visible watermark. That’s to say, Adv-watermark can also protect the copyright of the image because it carries the owner’s description. Therefore, Adv-watermark can accomplish two functions at the same time. This is a major advantage compared with the previous research. Specifically, peoples tend to share their images on social media to record their lives. They usually add a visible watermark to protect their copyright. But their images can also be identified and embezzled by malicious software. Adv-watermark can be used to avoid this situation. It not only protects the copyright of the image but also performs adversarial attacks to avoid being embezzled by malicious software. In this paper, we explore two kinds of media as the watermarks: logos and texts. Figure 2 lists the used watermarks, and some generated Adv-watermark examples are shown in Figure 1.

In summary, this paper has the following contributions:

1) We propose the Adv-watermark, a novel watermark perturbation for adversarial examples, which combines image watermarking techniques and adversarial example algorithms. Compared with the previous works, the proposed adversarial example is more realistic and effective.

2) We propose a novel optimization algorithm, which is called Basin Hopping Evolution (BHE), to generate adversarial examples efficiently. The proposed method adopts a population-based global search strategy to generate adversarial examples, and can achieve high performance in attacking DNN models.

3) Compared with the previous black-box attack methods, the proposed method can achieve a higher attack success rate. Moreover, the state-of-the-art image transformation defense methods can not defend the proposed attack method. The code is released at https://github.com/jiaxiaojunQAQ/Adv-watermark.git.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 introduces the details of the proposed Adv-watermark. Section 4 shows a series of experimental results and analysis. Finally, Section 5 gives the conclusion.

2 RELATED WORK

In this section, we investigate the attack methods and the visible watermarking methods.

2.1 Attack methods

In [6], Goodfellow et al. devise an effective method to calculate the adversarial examples, and the adversarial perturbation is generated according to the direction of the gradient change of the DNNs. This method is also called FGSM. Iterative FGSM (I-FGSM) [15] is an improved version of FGSM. I-FGSM constructs an adversarial example by multi-step and smaller movements, which greatly improves the success rate of the attack. The most common adversarial attack methods are under the $L_\infty$ and $L_2$ distance metric. But in [24], Papernot et al. propose to build adversarial saliency maps to generate adversarial examples under $L_0$ norm.
We use alpha blending in [31] to generate a visible watermark. Alpha channel (α channel) refers to the transparency of a foreground region w.r.t. the background image. In this paper, we use α to represent the value of the alpha channel, H to represent the host image whose size is $N \times M$, $W$ to represent the watermark image whose size is $n \times m$ and $G$ to represent the generated image with a watermark whose size is $N \times M$. When $i \in (p, p+n), j \in (q, q+m)$, the generation for $G$ is formulated as:

$$v(G)_{i,j} = (v(W)_{i-p,j-q} * \alpha + v(H)_{i,j} * (255 - \alpha))/255$$  \hspace{1cm} (1)

when $i \not\in (p, p+n), j \not\in (q, q+m)$, $G$ is formulated as:

$$v(G)_{i,j} = v(H)_{i,j}$$  \hspace{1cm} (2)

where $v(x)$ denotes the image $x$, the subscript $i,j$ of $v(x)$ represent the pixel position, and $p,q$ represent the position where the watermark image is embedded. As for the image watermark, we use UC Berkeley, CMU, MIT, Cambridge and Stanford University logo watermarks. Simultaneously, we also use the official ACM Multimedia logo from 2016 to 2020. As for text watermark, we use red, green, blue, black and gray fonts to generate adversarial examples. We also synthesize watermark images in different sizes to explore scale-aware effects. It is formulated as:

$$\eta = \min(W_h/s_l)/W_w, (H_h/s_l)/H_w),$$

\hspace{1cm} (3)

where $W_h$ and $H_h$ represent the width and height of the host image. $W_w$ and $H_w$ represent the width and height of the watermark image. $s_l$ is the scaling factor. And $W_{sw}$ and $H_{sw}$ represent the width and height of the scaled watermark image. Note that in this paper, we focus on the position and transparency of the watermark, not the rotation, etc.

### 3.2 Problem Formulation

We disguise adversarial noise as a visible watermark to achieve stealthiness. And the generation of adversarial examples is only related to the position and transparency of the watermark. Generating adversarial watermark images can be formalized as an optimization problem with constraints. The host image is assumed as $H$, the well-trained classification model is assumed as $f$ and the correct classification class of $H$ is $t$. $f_t(H)$ is the probability of $H$ belonging to the class $t$. Simultaneously, let $W$ be the watermark image and $g(H, W, p, q, \alpha)$ be the visible watermark algorithm. It embeds the watermark image $W$ in the position $(p,q)$ of the host image $H$. The $p, q$ and $\alpha$ are dependent on $W$, $H$, $f$. And the limitation of maximum transparency of the watermark is $L$. In the case of untargeted attacks, the goal of generation of adversarial examples can be transformed into finding the optimized solution $\varepsilon(p, q, \alpha)^\ast$. It is formulated as:

$$\min_{\varepsilon(p, q, \alpha)^\ast} f_t(g(H, W, p, q, \alpha))$$

subject to $\alpha \leq L$ \hspace{1cm} (4)

This problem involves two values: 1) the position $(p, q)$ of the watermark in the host image and 2) the transparency $\alpha$ of the watermark. Embedding the adversarial watermark which can be regarded as a practical perturbation into the host image modifies the local information of the host image. In this way, the adversarial watermark perturbation allows a clean image to be an adversarial example. Without affecting the visual effect of the image, the adversarial watermark disturbs the important local regions which determine the image classification to attack the well-trained classification model.
This is illustrated in Figure 3. From the heat-maps which are generated by Gradient-weighted Class Activation Mapping (Grad-CAM) [30], it is clear why the Resnet101 predicts the input images as the corresponding correct classes. And embedding the adversarial watermark into the image can modify the distribution of the maximum points on the generated heat-map.

### 3.3 Problem Solving

We propose a novel optimization algorithm, which is called Basin Hopping Evolution(BHE). The proposed method is a heuristic random search algorithm based on Basin Hopping, which can be used for finding the global minimum of a multivariate function. As shown in Figure 4, BHE includes Basin Hopping, crossover and selection operations. During each iteration, the current solutions (parents) use BH to produce a set of better solutions and conduct crossover operation to generate a new set of candidate solutions (children), And then in selection operation, compared with the corresponding parents to conduct, if the children are more suitable for the current population evolution (possess the smaller multivariate function value), they survive and are passed to the next generation.

#### 3.3.1 Population Initialization

BHE is an optimization algorithm based on group evolution. We regard each solution as an individual of a population. And the elements \((p, q \text{ and } \alpha)\) are considered as its genes. Let \(X_{i,g}\) denote the \(i\)-th individual in the \(g\)-th generation population. And \(X_{i,g,j} (j = 0, 1, 2)\) denote the \(j\)-th gene of \(X_{i,g}\). Therefore, we initialize a population as follows:

\[
X_{i,0,j} = X_{\text{min},j} + \text{rand}(0,1) \cdot (X_{\text{max},j} - X_{\text{min},j}); j = 0, 1, 2
\]  

(5)

where \(X_{i,0,j}\) is the \(j\)-th gene of the \(i\)-th individual in the initial population, \(X_{\text{min},j}\) is the minimum of the \(j\)-th gene and \(X_{\text{max},j}\) is the maximum of the \(j\)-th gene.

#### 3.3.2 Basin Hopping

Basin Hopping (BH) is a stochastic optimization algorithm. During each iteration, BH generates some new coordinates with random perturbations, next finds the local minimization, and finally accepts or rejects the new coordinates according to the minimized function value. We use BH to evolve a better individual \(V_{i,g}\) from \(X_{i,g}\).

In detail, \(f_t(g(H, W, p, q, \alpha))\) is assumed as \(f_t(\cdot)\). Starting with \(X_{i,g}\), a local optimal solution \(V_{i,g}\) of the function \(f_t(\cdot)\) is found by using a minimization method \(L(\cdot)\). Next we start the global search iterations and use \(\mu_g(X_{i,g})\) to represent the global neighborhood of \(X_{i,g}\). It is formulated as:

\[
\mu_g(X_{i,g}) = [X_{i,g}, X_{i,g} + r \cdot \bar{d}],
\]  

(6)

where \(\bar{d}\) is an \(n\)-dimensional Gaussian \((0,1)\) variable and \(r\) is a fixed step size. A new starting point is selected from the global neighborhood of \(V_{i,g}\). It is stored as \(V_{i,g}\). It is formulated as:

\[
V_{i,g} = G(\mu_g(V_{i,g}))
\]  

(7)

And then starting with \(V_{i,g}\), a local search is performed and the result is stored as \(S_{i,g}\). Finally, we use a function \(\text{Accept}(V_{i,g}, S_{i,g})\) to choose \(V_{i,g}\) or \(S_{i,g}\). And it is formulated as:

\[
\text{Accept}(V_{i,g}, S_{i,g}) = \begin{cases} 1 & f_t(S_{i,g}) \leq f_t(V_{i,g}) \\ 0 & f_t(S_{i,g}) > f_t(V_{i,g}) \end{cases}
\]  

(8)

The detail description is given in Algorithm 1. To represent BH algorithm simplify, it can be formulated as:

\[
V_{i,g} = \text{BH}(X_{i,g}, I),
\]  

(9)

where \(X_{i,g}\) represents the \(i\)-th solution in the \(g\)-th generation population, \(V_{i,g}\) represents the corresponding better solution using BH, \(\text{BH}(\cdot)\) represents the BH algorithm and \(I\) indicates the maximum number of Basin Hopping iterations which is a super parameter which we use a large number of experiments to certify.

#### Algorithm 1 BH algorithm

**Require:** The watermark image \(W\), the host image \(H\), the well-trained classifier \(f\) and \(X_{i,g}\)

**Ensure:** \(V_{i,g}\)

1. \(V_{i,g} = L(f_t(\cdot), X_{i,g})\);
2. **repeat**
   3. \(V_{i,g} = G(\mu_g(V_{i,g}))\);
   4. \(S_{i,g} = L(f_t(\cdot), V_{i,g})\);
   5. **if** \(\text{Accept}(V_{i,g}, S_{i,g})\) **then**
   6. \(V_{i,g} = S_{i,g}\);
   7. **end if**
3. **until** global stopping rule is satisfied
5. **return** \(V_{i,g}\)
3.3.3 Crossover. As for the current solution (parents) \( X_{i,g} \) and the corresponding BH optimization solution \( V_{i,g} \), we conduct crossover operation to get a candidate solution (child) \( U_{i,g} \). It is formulated as:

\[
U_{i,g,j} = \begin{cases} 
V_{i,g,j}, & \text{rand}(0, 1) \leq CR \\
X_{i,g,j}, & \text{others}
\end{cases}
\]

(10)

where \( U_{i,g,j} \) is the j-th gene of \( U_{i,g} \), \( V_{i,g,j} \) is the j-th gene of \( V_{i,g} \), \( X_{i,g,j} \) is the j-th gene of \( X_{i,g} \) and CR is the crossover probability which represents the degree of information exchange in the population evolution. It is a super parameter which we use a large number of experiments to certify.

3.3.4 Selection. We adopt a greedy selection strategy to select a better solution as the next generation solution. It is formulated as:

\[
X_{i,g+1} = \begin{cases} 
U_{i,g}, & f_{i}(U_{i,g}) \leq f_{i}(X_{i,g}) \\
X_{i,g}, & \text{others}
\end{cases}
\]

(11)

The detail description of BHE is given in Algorithm 2. And the generation process of the adversarial examples by using BHE is shown in Figure 5.

Algorithm 2 BHE algorithm

| Require: Population: M; Dimension: 3; Generation: N; Iteration: I; |
| Ensure: The best solution \( \cdot \Delta \) |

1: \( g \leftarrow 0; \)
2: for \( i = 1 \) to \( M \) do
3: \( \text{for } j = 1 \) to \( 3 \) do
4: \( X_{i,0,j} = \min_{j} + \text{rand}(0, 1) \cdot (\max_{j} - \min_{j}) \)
5: end for
6: end for
7: while \( f_{i}(\Delta) \geq \varepsilon \) and \( g \leq N \) do
8: \( \text{for } i = 1 \) to \( M \) do
9: \( \text{Basin Hopping} \)
10: \( V_{i,g} = \text{BH}(X_{i,g}, I) \)
11: \( \text{Crossover} \)
12: \( \text{for } j = 1 \) to \( 3 \) do
13: \( U_{i,g,j} = \text{Crossover}(V_{i,g,j}, X_{i,g,j}) \)
14: end for
15: \( \text{Selection} \)
16: \( \text{if } f_{i}(U_{i,g}) \leq f_{i}(X_{i,g}) \text{ then} \)
17: \( X_{i,g} = U_{i,g} \)
18: \( \text{if } f_{i}(X_{i,g}) \leq f_{i}(\Delta) \text{ then} \)
19: \( \Delta = X_{i,g} \)
20: end if
21: else
22: \( X_{i,g} = X_{i,g} \)
23: end if
24: end for
25: \( g = g + 1 \)
26: end while

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experiment Settings

We conduct experiments based on ImageNet [28] and CASIA-WebFace [37]. In detail, we randomly select 1,000 images from them to conduct the related experiments. We choose six classification models with different structures as threat models: Alexnet [14], VGG19 [32], SqueezeNet [11], Resnet101 [8], InceptionV1 [34] and InceptionV3 [35]. We also compare with other black-box attack methods to verify the proposed method: spatial attack [4], boundary attack [1], single-pixel attack [33] and pointwise attack [29]. As for these attack methods, we adopt their benchmark approaches and default parameters as recommended in Foolbox [27].

4.2 Optimization method implementation

The initial value of the step size \( r \) is set as 0.5. And the initial \( p, q \) and \( \alpha \) are set as 0, 0 and 100. The range of the \( p \) is \([0, W_h - W_{sw}]\). The range of the \( q \) is \([0, H_h - H_{sw}]\). And the range of the \( \alpha \) is \([100, 200]\).

4.3 Selection of hyper-parameters

We conduct a large number of experiments to determine two hyper-parameters in BHE. One is the number of basin-hopping iterations \( I \), the other one is crossover probability CR. We adopt BHE to attack DNN models using ACMMM 2020 logo with scale=1/4. In detail, we compute the attack success rates of the Resnet101 on 1000 random images of the ImageNet dataset. The result is shown in Table 1. From Table 1, it is clear that the attack success rate increases when \( I \) increases. That is, as the number of Basin Hopping iterations increases, the solution generated by BH will be better, resulting in achieving a higher attack success rate. But more iterations mean more time spent. Considering time complexity, we set CR to 0.9 and \( I \) to 3. In this way, Adv-watermark can achieve the highest attack success rate (60%). And in the original BH algorithm, the iteration \( I \) is set to 450.

4.4 Attack performance

In order to verify the proposed methods comprehensively, we choose five university logo watermarks and five official ACMMM watermarks as the image watermarks to generate corresponding adversarial examples. And we also choose five different color fonts as the text watermarks to generate corresponding adversarial examples. The average attack success rates of individual logos or text watermarks are reported in Table 2. The first column of each row shows the results of BH and the second column of each row shows the results of BHE. It is clear that the proposed BHE can achieve a high attack success rate. As for the university logo watermarks, when the watermark size is set as \( 4/9 \) of the host image size, the attack success rate can achieve about 97%. And when the watermark size is set as \( 1/16 \) of the host image size, the attack rate also can achieve 69%. As for the ACMMM logo watermarks, the average attack success rates
Table 2: The attack success rates of individual logo or text watermark.

| Logo/Text Watermarks | Alexnet | VGG19 | SqueezeNet1_0 | Resnet101 | InceptionV3 | Average |
|----------------------|---------|-------|----------------|-----------|-------------|---------|
| **ACMMM logo watermarks** |         |       |                |           |             |         |
| Scale=2/3            | 88%/92% | 77%/83% | 85%/88%       | 78%/83%   | 77%/79%    | 81%/85% |
| Scale=1/2            | 80%/88% | 69%/80% | 76%/82%       | 70%/78%   | 65%/74%    | 72%/80% |
| Scale=1/3            | 68%/76% | 54%/68% | 56%/69%       | 56%/66%   | 51%/61%    | 57%/68% |
| Scale=1/4            | 58%/69% | 43%/59% | 46%/62%       | 47%/58%   | 41%/52%    | 47%/60% |
| **Average**          | 74%/81% | 61%/72% | 66%/75%       | 63%/71%   | 59%/62%    | 65%/73% |
| **University logo watermarks** |         |       |                |           |             |         |
| Scale=2/3            | 96%/98% | 96%/96% | 95%/97%       | 96%/97%   | 96%/98%    | 96%/97% |
| Scale=1/2            | 90%/95% | 88%/90% | 88%/91%       | 88%/90%   | 87%/91%    | 89%/92% |
| Scale=1/3            | 78%/88% | 74%/76% | 73%/79%       | 72%/76%   | 68%/77%    | 73%/79% |
| Scale=1/4            | 66%/78% | 62%/66% | 61%/71%       | 60%/66%   | 54%/63%    | 61%/69% |
| **Average**          | 83%/90% | 80%/82% | 80%/84%       | 79%/82%   | 76%/82%    | 80%/84% |
| **Text watermarks**  |         |       |                |           |             |         |
| Font size=40         | 89%/91% | 82%/81% | 84%/85%       | 74%/76%   | 68%/73%    | 79%/81% |
| Font size=36         | 85%/89% | 79%/78% | 80%/83%       | 69%/73%   | 63%/69%    | 75%/78% |
| Font size=32         | 82%/85% | 75%/76% | 76%/80%       | 65%/69%   | 58%/65%    | 71%/75% |
| Font size=28         | 75%/80% | 70%/71% | 71%/75%       | 59%/66%   | 53%/60%    | 66%/70% |
| **Average**          | 83%/86% | 76%/76% | 78%/81%       | 67%/71%   | 61%/66%    | 73%/76% |

Table 3: The attack success rates with limit of embedded watermark position

| Watermark | Scale=1/4 | Scale=1/5 | Scale=1/6 | Scale=1/7 | Scale=1/8 |
|-----------|-----------|-----------|-----------|-----------|-----------|
| MIT logo  | 62%       | 58%       | 56%       | 55%       | 54%       |
| ACMMM2020 | 63%       | 59%       | 58%       | 57%       | 53%       |
| Red text  | 61%       | 57%       | 55%       | 53%       | 50%       |

Table 4: Comparison with other attack methods

| Network    | Spatial Attack | Boundary Attack | Single-Pixel | Pointwise Attack | SU logo | ACMMM2017 | Blue text |
|------------|----------------|-----------------|--------------|-----------------|---------|-----------|-----------|
| Resnet101  | 52%            | 37%             | 5%           | 7%              | 88%     | 75%       | 73%       |
| InceptionV3| 58%            | 48%             | 5%           | -               | 87%     | 72%       | 67%       |

Table 5: Performance on the state-of-the-art image transformation defense methods

| Network     | attacker | defender | Single-pixel Attack | Boundary Attack | CMU(1.5/2/3/4) | ACMMM2020(1.5/2/3/4) |
|-------------|----------|----------|---------------------|-----------------|-----------------|---------------------|
| Resnet101   | Jpeg     | defense  | 24%                 | 13%             | 100%/98%/94%/92%| 97%/95%/88%/93%    |
|             | Comdefend|         | 17%                 | 13%             | 99%/94%/88%/82% | 97%/94%/89%/82%    |
|             | HGD      |          | 42%                 | 34%             | 98%/95%/95%/94% | 97%/95%/92%/90%    |
| InceptionV3 | Jpeg     | defense  | 42%                 | 8%              | 100%/97%/94%/91| 99%/95%/90%/87%    |
|             | Comdefend|         | 34%                 | 8%              | 99%/95%/91%/86% | 98%/94%/90%/86%    |
|             | HGD      |          | 32%                 | 36%             | 98%/95%/89%/88% | 95%/90%/86%/85%    |

of them drop a little. That is because that the height-width ratio of the ACMMM watermark is not 1:1 (the height-width ratios of the ACMMM logo watermarks(2016-2020) are 1:2.6, 1:2.5, 1:3, 1:2.1 and 1:2.6), and the size of the ACMMM logo watermark is smaller than the university logo watermark when the scale is the same. In detail, when scale=1/4, the size of ACMMM2018 logo watermark is about 1/48 of the host watermark size. Even though the performance of the adversarial ACMMM logo watermarks declines a little, they also achieve a high attack success rate. Simultaneously, we use the text watermark to attack the well-trained classification models. As shown in Table 2, the proposed method can achieve about 86%, 76%, 81%, 71% and 66% average attack success rates on Alexnet, VGG19, SqueezeNet, Resnet101 and InceptionV3 with different font sizes. Compared with BH, the proposed BHE can
Table 6: Performance on the adversarial training

| Adversarial Training | MIT 1/4 | MIT 1/3 | ACMMM20 1/4 | ACMMM20 1/3 | Red Text 28 | Red Text 32 |
|----------------------|---------|---------|-------------|-------------|-------------|-------------|
| MIT (1/4)            | 50%     | 55%     | 74%         | 80%         | 91%         | 92%         |
| ACMMM20(1/4)        | 78%     | 83%     | 43%         | 48%         | 85%         | 86%         |
| Red Text(28)         | 71%     | 74%     | 72%         | 86%         | 44%         | 47%         |

Figure 6: (a) Limit of embedded watermark position. The face is in the red rectangle and the embedded watermark is restricted to the green rectangles. (b) Adversarial examples on the CASIA-WebFace dataset.

Figure 7: The adversarial examples with a variety of TV station logos. The original class labels are in black color and the class labels of the adversarial examples are in red color.

Figure 8: Layer-wise perturbation levels of the VGG16 model. Adversarial watermark and normal watermark added to clean images correspond to the $E_l$, respectively.

4.5 Comparisons with other attack methods

To quantitatively evaluate the proposed method performance, we compare the proposed method with other black-box attack methods: spatial attack [4], boundary attack [1], single-pixel attack [33] and pointwise attack [29]. In detail, we choose the SU and ACMMM2017 image watermarks with different scales and blue font text watermark with the different font sizes to complete the contrast experiments. Their average attack success rates are shown in Table 4. As shown in Table 4, it is clear that compared with other black-box attack methods, our attack method can achieve a higher attack success rate. In particular, the average attack success rate of SU reaches up to 88%.

In order to evaluate the robustness of the proposed method, we compare the Adv-watermark with other black-box attack methods: single-pixel attack and boundary attack, and choose three image transformation defense methods: Jpeg defend [3], Comdefend [12] and HGD [18]. From Table 5, it is clear that the existing image transformation defense methods are useful for single-pixel attack and boundary attack, but not useful for our proposed method. Compared with other attack methods, the proposed method is more robust. We also conduct adversarial training [20] to defend the proposed attack method. In detail, we inject adversarial examples generated by MIT, ACMMM2020 image watermark with scale $= 1/4$ and red text watermark with font $= 28$ into the original image dataset and retrain three Resnet101 on them respectively. And then we use these watermarks with different sizes to attack these models. The result is shown in Table 6. It is clear that the adversarial training cannot effectively defend Adv-watermark. Moreover, using another watermark to attack the adversarial training model can achieve a higher attack success rate. In other words, even though adversarial training increases the robustness to one watermark perturbation, it increases the vulnerability to another watermark perturbation.

4.6 Extension

The proposed method is not limited to using a watermark to generate an adversarial example. It can be extended to use the TV station logos to complete the attack. To make the generated adversarial
examples more realistic and imperceptible, we also choose more commonly used TV station logos to complete the attack. In detail, we select a variety of TV station logos, next limit the embedded position of the logos to the upper right corner of the host image and then use the proposed method to generate the adversarial examples. As shown in Figure 7, the generated adversarial examples are more realistic and common in the physical world.

4.7 Analysis for Adv-watermark

Compared with the previous attack methods, Adv-watermark pays more attention to generate realistic adversarial examples. We find DNN models are spatially vulnerable, which adding perturbations at a specific position to clean images can attack them easily. To investigate this characteristic, we conduct a comparative experiment to evaluate layer-wise perturbations of the VGG16 model fed adversarial watermark images and normal watermark images, respectively. The difference between normal watermarks and adversarial watermarks is that they are positioned differently on clean images. The perturbation level in layer $l$ can be formulated as:

$$E_l (x_w, x) = \frac{\|f_l (x_w) - f_l (x)\|_2}{\|f_l (x)\|_2},$$

where $x$ represents a clean image, $x_w$ represents the clean image with adversarial or normal watermark and $f_l (\cdot)$ represents the $l$-th layer of the VGG16 model.

The result is shown in Figure 8. The red curve represents the $E_l$ for adversarial watermark perturbations and the blue curve represents the $E_l$ for normal watermark perturbations. Specifically, the red curve is the average result on 30 randomly picked images with the adversarial watermarks and the blue curve is the average result on 30 same images with the normal watermarks. It is clear that the watermark perturbation is progressively enlarged with the layer hierarchy. But in the top layer, the adversarial watermark perturbation is much higher than the normal watermark perturbation. Because the classification result is dependent on the top-level features, the adversarial watermark perturbation can fool DNN models but the normal watermark perturbation can not.

5 CONCLUSION

In this paper, we discovered DNN models were spatially vulnerable, which adding perturbations at a specific position to clean images could attack models easily. And then we proposed a novel attacking method which used the real watermark to attack the well-trained classifier. Our adversarial perturbation was meaningful, which was different from the traditional ones. We formulated the watermark attack problem as a global optimization problem, and proposed a novel optimization algorithm (BHE) to generate adversarial examples. Compared with the previous BH, BHE achieved a higher attack success rate. Moreover, the Adv-watermark was more robust, because the image transformation defense methods could not defend the proposed attack method. And the proposed method could be more commonly used in the real world.

ACKNOWLEDGEMENT

Supported by the National Key R&D Program of China under Grant 2018AAA0102503, National Natural Science Foundation of China (No. U1936210, U1736219, 61971016, 61806109), The Open Research Fund from Shenzhen Research Institute of Big Data, under Grant No. 2019ORF01010, Beijing Natural Science Foundation (No.L182057), Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004.
