Towards Interpreting Vulnerability of Multi-Instance Learning via Customized and Universal Adversarial Perturbations

Yu-Xuan Zhang\textsuperscript{a}, Hua Meng\textsuperscript{a}, Xue-Mei Cao\textsuperscript{b}, Zhengchun Zhou\textsuperscript{a,}{\textsuperscript{c}}, Mei Yang\textsuperscript{a}, Avik Ranjan Adhikary\textsuperscript{a}

\textsuperscript{a}School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611730, China
\textsuperscript{b}School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu, 611130, China
\textsuperscript{c}School of Computer Science, Southwest Petroleum University, Chengdu 610500, China

Abstract

Multiple-Instance Learning (MIL) is a recent machine learning paradigm which is immensely useful in various real-life applications, like image analysis, video anomaly detection, text classification, etc. It is well known that most of the existing machine learning classifiers are highly vulnerable to adversarial perturbations. Since MIL is a weakly supervised learning, where information is available for a set of instances, called bag and not for every instances, adversarial perturbations can be fatal. In this paper, we have proposed two adversarial perturbation methods to analyze the effect of adversarial perturbations to interpret the vulnerability of MIL methods. Out of the two algorithms, one can be customized for every bag, and the other is a universal one, which can affect all bags in a given data set and thus has some generalizability. Through simulations, we have also shown the effectiveness of the proposed algorithms to fool the state-of-the-art (SOTA) MIL methods. Finally, we have discussed through experiments, about taking care of these kind of adversarial perturbations through a simple strategy. Source codes are available at [https://github.com/InkiInkt/MI-UAP](https://github.com/InkiInkt/MI-UAP).

Keywords: Customized perturbation, Multi-instance learning, Universal perturbation, Vulnerability

1. Introduction

The data structure of multi-instance learning (MIL) makes it unique by describing each sample as a bag of many instances. In MIL, the number of instances is typically higher than that of bags, while the supervised information is only provided at the bag-level. According to the basic MIL assumption \cite{5}, a bag is positive if it contains at least one positive instance; otherwise it is negative. Such weakly supervised scenarios are challenging since they only offer a very small amount of information to support decision-making. But, this makes MIL more attractive to the researchers, which enables them to work at a higher level (bag-level) and address many real-world issues, which includes \cite{16, 22, 37, 39}, drug production \cite{5, 12, 32}, and video abnormal detection \cite{16, 22, 29}.

MIL models can be broadly classified into two categories \cite{14}: embedding- and instance-based methods. The basic difference between these two approaches is that, the former obtains the label of the bag, end-to-end, without the need to provide or learn an instance-level classifier. As a result, this type of method has gained popularity among researchers in recent times, which gives rise to two main ideas: neural network \cite{2, 8, 10, 28} and traditional embedding methods \cite{31, 36}. The main task of these methods is to learn the bag embedding through a permutation-invariant transformation, so the quality of the embedding directly affects the classification performance. However, these methods hardly take algorithm security into account, which makes the algorithm seriously vulnerable to adversarial perturbations, which can fool image search engines or surveillance cameras.

To interpret the vulnerability of MIL algorithm, in this paper, we propose multi-instance customized and universal adversarial perturbation (MI-CAP and MI-UAP), a demonstration is shown in Figure \ref{fig:1}. For a given bag containing instances, we have proposed two ways to generate perturbations. First, MI-CAP is used to fool the prediction of the MIL learner of the current bag by iterating the gradient-related computation. The perturbation that is created for each bag may be unique, which is also the meaning of “customized”. In the next method, MI-UAP obtains a universal perturbation that is easy to store and can be easily generalized by repeatedly fine-tuning the perturbation generated by MI-CAP. While such a process will result in some performance losses, since it needs to store one perturbation, it is very advantageous when dealing with data sets. To verify the effectiveness of these methods, we have conducted experiments on three well-known image databases, five classic MIL data sets, and two popular video anomaly detection data sets. The fooled methods are all state-of-the-art (SOTA) MIL methods, including five neural networks and five traditional methods. The experimental results expose the vulnerability of these methods, which can be fatal in some circumstances, especially when the bag actually does include instances of our demand. Hence, we also propose a simple experiment to illustrate the prevention of the proposed adversarial perturbations.
In summary, our main contributions are as follows:

- We design two strategies to generate perturbations to expose the vulnerability of SOTA MIL methods, one of which is bag-customized and the other indirectly demonstrates the existence of universal bag-agnostic perturbations for these MIL methods.

- The proposed methods and the generated adversarial bags can be considered useful knowledge or prior experience for future learners. The reason is that MI-CAP can generate perturbations that are unaffected by other bags, while the perturbations generated by MI-UAP can be easily stored as part of the knowledge base.

- We demonstrate the generalizability of these perturbations in experiments, showing that perturbations generated by one neural network can fool other networks as well. In addition, we propose a simple strategy to mitigate the effect of these perturbations as much as possible.

2. Related Work

In this section, we revisit some works related to MIL methods and adversarial perturbation approaches.

2.1. MIL Methods

In his study in 1997 about drug activity prediction [5], Dietterich et al. proposed MIL. MIL is currently used in many real-world applications, such as image classification and video anomaly detection due to its unique features, and it is can be broadly classified in two branches. For traditional methods, MILES [3] builds a permutation-invariant transformation that embeds the bag into a new feature space for instance-level classifiers to function. BACIC and ELDB [41, 36] use the set-kernel as the foundation, and each feature of the bag embedding vector denotes how similar it is to the corresponding key bag. miVLAD and miFV [31] are very similar to BACIC and ELDB, with the primary distinction being that the instance-set distance metric takes the place of the set-kernel. Additional methods of this type include BDR [7], miVLAD [31], ISK [33], AEMI [37], MSK [38].

At the same time, MIL neural networks also advanced quickly, particularly the more recent attention-based ones. ABMIL [8] presents two attention strategies by expressing the MIL problem as learning the Bernoulli distribution of the bag label, where the bag label probability is completely parameterized by neural networks. LAMIL [25] proposes a new loss function based on the attention mechanism, which completes the prediction of the bag label while calculating the weight and uses the consistency cost to boost the generalization performance of the model. DSMIL [14] introduces a novel MIL aggregator to model the relationships between the instances in the bag and employs self-supervised contrastive learning to extract the good representation of bags. It also adopts a pyramidal mechanism to further enhance classification performance. MAMIL [10] takes into account nearby instances of each analysis instance to efficiently process different types of instances and implement a diverse feature representation of the bag. Some other methods include MIL-RNN [2], C2C [24], BSN [29], DP-MINN [35] and BP-MIP [40].

2.2. Adversarial Perturbation Approaches

Numerous methods have demonstrated the vulnerability of traditional machine learning or deep learning algorithms. DeepFool [21] shows how deep networks can be fooled with only minor, well-sought perturbations to images. UAP [20] proves the existence of an image-agnostic and tiny perturbation vector that causes natural images to be misclassified frequently. S4 presents a methodical and comprehensive overview of the main threats of attacks for three most popular data types, including images, graphs and text. Homotopy-Attack [44] fools the neural networks by jointly addressing the sparsity and the perturbation bound in one unified framework. UAP-GAN [30] respectively studies the robustness of image classification and captioning systems based on convolutional neural networks and recurrent neural networks. LFT [42] leverages bilevel optimization and learning-to-optimize techniques for perturbation generation with an improved attack success rate.

In summary, MIL is a learning form with a special data structure that has derived various learning strategies and achieved good results in several real-life applications, like image classification, video anomaly detection, and others. However, since
MIL is a special kind of machine learning, which is vulnerable to adversarial perturbations, the security and vulnerabilities of MIL algorithms need to be discussed.

3. Methodology

In this section, we will firstly present the problem statement and then propose two perturbation strategies for MIL, namely the customized and universal adversarial perturbations.

3.1. Problem Statement

In the MIL scenario, each bag \( B_i = \{x_{i1}, x_{i2}, \ldots, x_{in_i}\} \) corresponds to a sample, where \( x_{ij} \in \mathbb{R}^d, j = 1, 2, \ldots, n_i \) is the \( j \)-th instance of \( B_i \), \( n_i \) is the cardinality of \( B_i \), and \( d \) is the dimension. Based on the basic MIL assumption \cite{1}, the supervised information \( y_j \in Y = \{1, 0\} \) of the label of \( B_i \) is only given at the bag-level, while instance labels are either unknown or unavailable. In addition, a bag is positive while instance labels are either unknown or unavailable. Each instance \( x_{ij} \) can have an assumed label \( y_{ij} \in Y \) from which the bag’s label can be deduced:

\[
y_i = \begin{cases} 
1, & \forall j \in 1, 2, \ldots, n_i, \exists y_{ij} = 1; \\
0, & \text{otherwise}.
\end{cases}
\]

One of the most popular MIL research topics is neural network models. Take as an example these methods \cite{6, 10, 25}, which aim to learn an end-to-end mapping \( F : B_i \rightarrow \hat{y}_i = \sigma(b_i) = \arg \max_r p_r \) to embed each bag into a new feature space while keeping the highest degree of distinguishability to maximize the classification performance, where \( p_r \in \mathbb{R} \) is the probability that the bag is from class \( c \) and \( f() \) a fully connected layer as well as \( b_i \in \mathbb{R}^d \) is the embedding vector of \( B_i \).

However, they are easily fooled by a slight perturbation \( \epsilon \) into excluding a bag from the decision region it belongs to. This is exactly what we intend to do, i.e., expose the MIL algorithms’ vulnerability by maximizing the fool rate:

\[
\max_{\epsilon} \frac{\sum_{i \in S} I(\hat{y}_i \neq \hat{y}_i)}{\sum_{i \in S} I(\hat{y}_i = \hat{y}_i)}, \tag{2}
\]

where \( S = \{B_i\}_{i=1}^N \) is the data set with the size \( N \), \( \epsilon \) is a vector of the same dimension as \( x_{ij} \), and \( I(\cdot) \) is an indicator function that outputs 1 if the input is true, otherwise \( I(\cdot) = 0 \). Here \( \hat{y}_i = f(B_i \oplus \epsilon) \), and \( B_i \oplus \epsilon \) refers to adding interference to each instance in the bag, which is equivalent to \( \{x_{ij} + \epsilon\}_{i=1}^{n_i} \). This has two advantages: 1) “Usually unequal number of instances in different bags” is not a restriction; and 2) The number of positive instances is usually much smaller than the number of negative ones, while adding perturbations to positive ones takes extra time. This problem is avoided by Eq. (2). Next, we will focus on two perturbation generation methods.

### 3.2. Customized Adversarial Perturbation (CAP)

It is not uncommon to perturb a specific sample \cite{6, 21, 34}, and there is a clear advantage to doing so: Ideally, a particular perturbation can always be generated to lead the classifier to misclassify the current sample. Like their basic idea, the goal of multi-instance learning customized adversarial perturbation (MI-CAP) is to fool the MIL learner by generating a unique perturbation \( \epsilon_i \) for \( B_i \) over a number of iterations \( L \). Such a process is called “customized” because the same perturbations may not exist at all for a given data set. Specifically, the minimal perturbation that fools the classifier to mispredict the label of \( B_i \) can be written as

\[
\arg \min_{\epsilon_i} \|\epsilon_i\|_2 \quad \text{s.t.} \quad p_{B_i}^{true} \geq \hat{y}_i, \tau_i = Y \setminus \{\hat{y}_i\} \tag{3}
\]

where \( p_{B_i}^{true} \in l(b_i^{true}) \) and \( b_i^{true} \) is the embedding vector of the perturbed bag \( B_i \oplus \epsilon_i \). Algorithm \cite{1} gives the solution to this equation based on the core idea of DeepFool \cite{21}, which can handle non-convex optimization problems brought by neural networks as the classifier.

Lines 1–2 are the initialization of MI-CAP. This stage initializes the perturbation and the gradient vector, and get the pre-

#### Algorithm 1: MI-CAP

**Data:**
- Bag \( B_i \);
- MIL neural network \( F \);
- Number of iterations \( L_i \);
- Gradient computation mode: “ave” or “att”;

**Result:**
- Customized perturbation \( \epsilon_i \);

1. Initialize the perturbation \( \epsilon_i^l = 0 \) and the gradient vector \( g_i^l = 0 \), \( l = 0 \;
2. Get the prediction \( \hat{y}_i = F(B_i + \epsilon_i^l) \);
3. while \( F(B_i \oplus \epsilon_i^l) = = F(B_i) \) and \( l < L_i \)
   do
   4. \( l++ \);
   5. Compute the gradient \( \hat{G}_i = \frac{\partial p_i}{\partial B_i \partial \epsilon_i^l} \);
   6. Compute the gradient \( \hat{G}_i = \frac{\partial p_i}{\partial B_i \partial \epsilon_i^l} \), where \( \tau_i = \) Eq. (4);
   7. if “ave” is used then
      8. \( g_i^l = \frac{1}{n_i} \sum_{j=1}^{n_i} \hat{g}_j \), where \( \hat{g}_j \) is the \( j \)-th row of \( \hat{G}_i \);
      9. \( g_i^l = \frac{1}{n_i} \sum_{j=1}^{n_i} \hat{g}_j \), where \( \hat{g}_j \) is the \( j \)-th row of \( \hat{G}_i \);
   else
      10. \( g_i^l = \hat{g}_i \), where \( i \) = Eq. (4);
      11. \( g_i^l = \hat{g}_i \);
   end
12. \( \Delta \epsilon_i = g_i^l \oplus g_i^{l-1} \);
13. \( \Delta \epsilon_i = |\eta \epsilon_i| \frac{\Delta \epsilon_i}{\|\Delta \epsilon_i\|} \), where \( \eta > 0 \) is a small number used to avoid division by 0;
14. Update \( \hat{y}_i \) via \( F(B_i \oplus \epsilon_i^l) \);
15. Update \( \epsilon_i^l = \epsilon_i^{l-1} + \Delta \epsilon_i \);
16. Update \( \epsilon_i^l \);
17. End
18. return \( \epsilon_i^l \);
gradient, together with fine-tuning the \( \epsilon \) estimation. Its critical steps are the computation and updating of the ability \( p \) of \( \tilde{y} \) to predict the bag label \( \tilde{L} \). Perturbation within a finite number of iterations while fooling the MIL learner by making the bag deviate from its original classification region. A schematic representation of the strategy of DeepFool is based on identical-resolution images, whereas the positive region \( A^+ \) is not necessarily equal. But, letting \( A^+ \) be much larger than the other values in \( A \), it should be emphasized that the perturbation generation process would be lost. That is, we need some additional operations to unify all gradients into the same shape since the ones corresponding to different bags usually have different shapes.

Therefore, we offer two solutions to the above problem: The first averages each row of \( \mathbf{G}_i \) and \( \mathbf{G}_j \), as in Lines 8–9, and the second skillfully makes use of the attention-based MIL methods \cite{8, 10}, as in Lines 11–12. Specifically, such methods can return the attention weight \( A_i = [\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{in}] \) related to \( B_i \) while obtaining \( \tilde{y}_i \), where \( \alpha_i \) is the attention value of the instance \( x_{ij} \). \( A_i \) reflects how much the instances in the bag contribute to the prediction. Taking the example of having only one positive instance in the bag, in an ideal case, one of the values in \( A_i \) might be much larger than the other values in \( A_i \). To test this conjecture, the attention values of ABMIL on the MNIST data set with 50 training epochs are gathered, i.e., \( \mathcal{A} = \bigcup_{i=1}^{N} A_i \). The specific parameter settings will be explained in Section 4.1. The distribution and the kernel density estimation (KDE) results of \( \mathcal{A} \) are shown in Figure 3. The results show that the vast majority of instances have attention values that are close to zero, suggesting the gradients associated with them may not be as significant.

As a result, \( A_i \) can be utilized to assist in identifying the instance in which the current bag, when left undisturbed, contributes the most to the bag label. The index of such an instance is

\[
i_t = \arg \max_j \alpha_i j \tag{4}\]

and only the gradient of the instance indicated by \( i_t \) will be used when updating \( \epsilon_t \).

### 3.3. Universal Adversarial Perturbation (UAP)

MI-UAP is an extension of MI-CAP that iteratively utilizes the perturbations produced by the latter to obtain a universal perturbation. This necessarily sacrifices some performance since customized perturbations always try to fool the MIL learner into making an incorrect judgment about each bag. But the universal perturbation also implies flexibility, generalizability, and ease of storage, which may have greater applications in the future, such as intrinsic knowledge obtained from the data set. \cite{20, 30, 42} and others also established the existence of such a perturbation based on traditional neural networks.
Algorithm 2: MI-UAP

Data:
- Bag $B_i$;
- MIL neural network $\mathcal{F}$;
- Number of iterations $L_1$ and $L_2$;
- Gradient computation mode: “ave” or “att”;
- Fooling rate threshold $\delta$;
- Magnitude control parameter $\xi$;

Result:
- Universal perturbation $\epsilon$;

1. Initialize the perturbation $\epsilon' = 0$ and the fooling rate $r' = -1$, where $l = 0$;
2. while $r' < \delta$ and $l < L_2$ do
3.     $l++$;
4.     for $B_i \in \mathcal{S}$ do
5.         if $\hat{y}_i = \hat{y}_i$, then
6.             $\epsilon' = \text{MI-CAP}(B, \mathcal{F}, L_1)$;
7.             $\Delta \epsilon' = \text{Eq. (5)}$ with $\epsilon'$;
8.             $\epsilon' \leftarrow \epsilon' + \Delta \epsilon'$;
9.             Project $\epsilon'$ via Eq. (7) with $\xi$;
10.      end
11.     end
12.     $r' = \text{Eq. (2)}$;
13.     if $r' > r^i$ then
14.         $r^i = r'$;
15.         $\epsilon^0 = \epsilon'$;
16.     end
17. end
18. return $\epsilon'$;

Therefore, we aim to find a universal perturbation $\epsilon$ that satisfies:

$$\arg \min_{\epsilon} \|\epsilon\|_2, \quad \text{s.t.} \quad \frac{E_{\mathcal{F}}(\epsilon')}{\|\epsilon\|_\infty} \geq \delta; \quad \|\epsilon\|_\infty \leq \xi, \quad (5)$$

where $\delta > 0$ is the fooling rate threshold parameter and $\xi > 0$ is the magnitude control parameter of $\epsilon$. Similar to MI-CAP, a schematic representation of MI-UAP is shown in Figure 2(b) and its solution is shown in Algorithm 2.

Line 1 starts MI-UAP by setting the universal perturbation $\epsilon^0 = 0$ and the fooling rate $r^0 = 0$.

Lines 2–17 correspond to the entire optimization process, which ends when the predetermined threshold $\delta$ or the number of iterations $L_2$ is reached. In the algorithm’s main loop, we will iterate over each bag $B_i$ in the data set $\mathcal{S}$, as finding the smallest perturbation $\epsilon_i'$ for each bag is one of the critical phases. This is insufficient, though, as we require a perturbation that is as effective for all bags as possible. As a result, using $\epsilon_i'$ as the basis, we need to find an extra perturbation $\Delta \epsilon_i$ that satisfies:

$$\Delta \epsilon_i = \arg \min_{\varsigma} \|\varsigma\|_2, \quad \text{s.t.} \quad \mathcal{F}(B_i \oplus (\epsilon_i' + \varsigma)) \neq \hat{y}_i. \quad (6)$$

Once we have $\Delta \epsilon_i$, we will utilize it to update $\epsilon_i'$ as stated in Line 8. In addition, the updated perturbation might not satisfy the constraints in Equation (5), for which we need to project it into the new feature space:

$$\epsilon_i' = \arg \min_{\varsigma} \|\epsilon_i' - \varsigma\|_2, \quad \text{s.t.} \quad \|\varsigma\|_2 \leq \xi, \quad (7)$$

Note that Eq. (7) is also used to project the perturbation generated by MI-CAP to adjust its magnitude. At last, the perturbation $\epsilon$ with the highest fooling rate $r$ is returned.

3.4. Discussion and Extension

For a more in-depth investigation of MI-CAP and MI-UAP, we here will discuss their time complexities and extend them to more complex cases with the bag of images.

Proposition 1. The time complexity for MI-CAP and MI-UAP to generate perturbations for all bags is $O(d^2 \Sigma)$ and $O(n d^2 \Sigma)$, respectively, where $d$ is the dimension, $\Sigma = \sum_i n_i$ is the total number of instances, and $N$ is the number of bags.

Proof. For MI-CAP, the gradient calculation in its main loop, which is directly tied to the complexity of the network structure given $\mathcal{F}$, determines the majority of the time complexity of this algorithm. Using DSMIL [14] as an example, which has a more complex network structure, it mostly consists of linear transformations and convolution operations. DSMIL therefore costs $O(n d^2 L_3)$ to handle a bag. Lines 14–16 are primarily composed of simple addition and subtraction operations, whereas Line 17 solely employs the pretrained $\mathcal{F}$ to predict the labels of perturbed bags $B_i \oplus \epsilon_i$. When contrasting these costs with $n d^2$, they can be disregarded. Generally, we have $L_1 \ll \Sigma$, where $\Sigma = \sum_i n_i$. Therefore, the time complexity of MI-CAP applied to the whole data set is $O(d^2 L_1 \sum_i n_i) = O(d^2 \Sigma)$. For MI-UAP, its main loop is itself an operation on all bags, which costs $O(n d^2 L_2 \Sigma)$ and can be simplified to $O(N d^2 \Sigma)$. Despite taking longer than MI-CAP to generate the perturbation, MI-UAP only costs $O(1)$ during inference to a given bag rather than $O(n d^2 L_1)$.

The perturbation generation of MI-CAP and MI-UAP is carried out on the premise that the instance in the bag is a vector, which does not satisfy the more complex classification of the bag structure mentioned in [8,10]. It is therefore necessary to extend these two algorithms to enhance their adaptability in the above scenarios.

Let $\mathcal{B}_i = \{I_{j1}, I_{j2}, \ldots, I_{jm}\}$ be a bag, where $I_{ij}$ is the $j$-th image of $\mathcal{B}_i$. The objective in Eq. (2) will be rewritten as

$$\max_{B_i \in \mathcal{B}} \sum_{j=1}^{m} \frac{\|\mathcal{F}'(B_i \oplus \epsilon') \neq \hat{y}_i\|}{\sum_{j} \|y_i \neq \hat{y}_i\|}, \quad (8)$$

where $\mathcal{B}_i \oplus \epsilon' = \{I_{j1} + \epsilon', I_{j2} + \epsilon', \ldots, I_{jm} + \epsilon\}$ and $\epsilon'$ has the same amount of rows and columns as $I_{ij}$. Here, $\mathcal{F}'$ denotes the addition of a convolutional block to the network in order to learn the vector representation of the image, and the specific structure is taken from [8].
4. Experiments

In this section, we will utilize MI-CAP and MI-UAP to expose the vulnerability of the SOTA MIL neural networks, which are conducted on three image databases, two VAD data sets, and five benchmark data sets.

4.1. Experimental Setups

Five networks were selected to evaluate our methods, including ABMIL and GAMIL [8], LAMIL [25], DSMIL [14], and MAMIL [10]. With the exception of the training epoch and learning rate, we exactly follow the original paper’s description for their parameter setups. For consistency, these two parameters are set to be 50 and 0.0001 for all algorithms, respectively. For MI-CAP and MI-UAP, we set $L_1 = 10$, $L_2 = 50$, $\delta = 0.5$, and $\xi$ was enumerated in $\{0.01, 0.05, 0.1, 0.2, \ldots, 0.5, 1\}$. The modes they use are denoted by “ave” and “att”. The reason we set $\delta$ this way is that, assuming the data set is class-balanced at this point, only half the fooling rate is required to fool the judgement of the MIL learner. And the reason for $\xi$ is that we normalize all of the data sets, so the setting of it indirectly reflects how perturbation and data space are related. And we would select an appropriate $\xi$ before comparing performance, as a too large $\xi$ will distort the bags too much and render the perturbation worthless.

Three famous image databases were used to evaluate the effectiveness and the generalizability of our methods, including MNIST [13], CIFAR10 [11], and STL10 [4]. These databases are not applicable to MIL scenarios and require some special strategies to convert into data sets with MIL data structures. Thanks to the contribution of [8] in this regard, the following are the steps to generate MIL data: 1) Specify the number of bags, the range of bag sizes, and the target class of the database (here we set the target class to 9). The image from the target class will be used as the positive instance, which determines the label of the bag; 2) Generate the size of each bag at random, then choose the required number of images from the database. If an image from the target class is present, the label is positive; otherwise, it is negative; and 3) Return the data set once sufficient bags are generated.

The data sets produced with the above three image databases closely mimic the MIL application scenarios. However, these data sets were only generated based on the MIL assumption, and the actual application scenarios may be much more complex. In this regard, we employ two VAD data sets to validate the proposed algorithms. Specifically, ShanghaiTech [19] [43] and UCF-Crime [26] are a medium-scale data set from fixed-angle street video surveillance and a large-scale data set from real-world street and indoor surveillance cameras. Because there are typically many frames in a single video and supervision information is only provided at the video-level, VAD is a typical weak supervision problem. Therefore, we use the setting of [27], split each video in the aforementioned data sets into multiple clips, treat each clip as a bag, and use I3D [9] for preprocessing to help the model learn video features more effectively.

Furthermore, MI-CAP and MI-UAP cannot be used to directly fool traditional MIL methods because most of them do not involve gradient computation but rather certain methods based on key samples, support vectors, or others. So, we use the universal perturbation to attack traditional MIL methods, including BAMIC [41], miFV and miVLAD [31], ELDB [36], and AEMI [37]. The data sets used both are well-known benchmark drug activity prediction [5] and image classification data sets [1] [15]. The parameter settings of these five methods are referenced to [37].

4.2. Parameter Analysis

Figures 1 shows the parameter analysis experiments for the crucial parameter $\xi$ in MI-CAP and MI-UAP. The abscissa represents $\xi$, and the ordinate represents the recall value of the classification. The results show that the MIL networks may be fooled more readily as perturbation strength increases and can even drop recall near zero when perturbation reaches a certain threshold. This implies that perturbations with high fooling rates are entirely possible, right? Evidently not, and Figure 5 clearly refutes the claim that “perturbations with large fooling rates are also more effective.” It can be seen from this figure that the image will be distorted when the perturbation exceeds a certain magnitude, which is obviously unreasonable. Therefore, we recommend that the value of $\xi$ for image classification not be more than 0.2. Figure 1 further demonstrates that MI-CAP (ave), which is followed by MI-CAP (att), has the best overall performance, while MI-UAP (ave) only gains an edge over others on ABMIL. Meanwhile, a very interesting case is that MI-CAP (ave) gets similar results on GAMIL for $\xi = 0.01$ and $\xi = 0.2$, while the former has a significantly smaller perturbation magnitude than the latter. This partially reflects its potential, which we will focus on expanding in the future. Some other findings are as follows: a) The perturbation generated by MI-CAP and MI-UAP is not random or irregular, and it can show a better effect with an increase in the perturbation magnitude without taking the distortion of the bags into account; Perturbation generation mode ave outperforms mode att overall, probably because its generated perturbations are added to each instance in the bag, while mode att only focuses on the gradient of the most important instances; c) The original bag is only slightly perturbed when $\xi \leq 0.05$; and d) Perturbations affect both target and irrelevant background areas, illustrating the need to limit the regions in which they work, something that current algorithms are unable to do.

4.3. Effectiveness Comparison

Table 1 displays the findings of the effectiveness comparison of our methods against the “random” and “mean” approaches introduced in [20]. “No” specifically refers to no perturbation added, which corresponds to the original classification performance of the MIL neural network. For each data set under the metric “No”, the average accuracy (acc) and recall (rec) of five individual experiments under $\xi = 0.2$ were given, together with their standard deviation (the value with “±”), e.g.,

$$Acc = \frac{1}{N} \sum_{i=1}^{N} I(y_i = y),$$

(9)
For others, we show a decrease in the algorithm accuracy after adding perturbation, e.g.,

$$\text{Decrease} = \text{Acc} - \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i = \bar{y}_i).$$ \hfill (10)

For example, the second and third rows of the fourth column indicate that the average accuracy of ABMIL on the MNIST data set is 95.2%, which drops by 4.4% after adding random perturbation. One very important reason is that the accuracy after perturbation cannot fully reflect the performance of the perturbation algorithm because the results of multiple independent experiments of the method without perturbation are typically not equal. The down arrow indicates the best perturbation algorithm for the current column, and the up arrow denotes that the perturbation improves model accuracy. Note that the recall for more than three algorithms is below 0.4 on CIFAR10 under “No”, hence it is not displayed here.

The results show that MI-CAP and MI-UAP can fool the MIL neural network to a large extent, even bringing the recall on some data sets below 0.6. This is sufficient for MIL because adding perturbations to make the negative bag be predicted as positive is inherently ridiculous. Another good finding is that the algorithm can be very unstable when predicting data with perturbations, which will further lower the confidence of the algorithm. Meanwhile, the results in Table 1 and Figures 4 and 5 comprehensively reveal that our MI-CAP and MI-UAP are significantly different from random or mean perturbations, in terms of both performance and the regularity they exhibit.

VAD is one of the crucial applications that is directly tied to people’s work and daily lives. Criminals might inflict significant property damage if they adopt some techniques to interfere with the process, such as generating specialized perturbations to fool video surveillance. To think more deeply about adversarial methods, we conducted experiments on such data sets, as shown in Table 2. It should be observed that the five MIL neural networks perform poorly on CIFAR10 and STL10; in particular, the recall value on CIFAR10 is very low, making it difficult to accurately depict MI-CAP and MI-UAP’s performance. There-
Table 1: Performance comparison for network-based classification on three databases using seven perturbation strategies ($\xi = 0.2$).

| Data set | Evaluator | Perturbation | ABMIL [8] | GAMIL [8] | LAMIL [25] | DSMIL [14] | MAMIL [10] |
|----------|-----------|--------------|-----------|-----------|------------|-------------|-------------|
| MNIST    | No        | MI-CAP (ave) | 0.952 ± 0.11 | 0.972 ± 0.18 | 0.960 ± 0.14 | 0.956 ± 0.17 | 0.968 ± 0.11 |
|          | Random    | MI-CAP (ave) | 0.044 ± 0.41 | 0.088 ± 0.03 | 0.104 ± 0.09 | 0.076 ± 0.40 | 0.048 ± 0.24 |
|          | Mean      | MI-CAP (ave) | 0.020 ± 0.30 | 0.056 ± 0.36 | 0.040 ± 0.02 | 0.076 ± 0.14 | 0.140 ± 0.39 |
|          | Acc       | MI-CAP (ave) | 0.224 ± 0.65 | 0.496 ± 0.86 | 0.364 ± 0.71 | 0.492 ± 0.75 | 0.312 ± 0.13 |
|          | Rec       | MI-CAP (ave) | 0.320 ± 0.039 | 0.124 ± 0.091 | 0.112 ± 0.061 | 0.020 ± 0.26 | 0.156 ± 0.204 |
|          |           | MI-CAP (att) | 0.060 ± 0.039 | 0.144 ± 0.059 | 0.088 ± 0.039 | 0.144 ± 0.108 | 0.020 ± 0.030 |
| CIFAR10  | No        | MI-CAP (ave) | 0.700 ± 0.35 | 0.640 ± 0.37 | 0.624 ± 0.043 | 0.676 ± 0.030 | 0.680 ± 0.020 |
|          | Random    | MI-CAP (ave) | 0.112 ± 0.050 | 0.024 ± 0.022 | 0.004 ± 0.020 | 0.076 ± 0.014 | 0.120 ± 0.068 |
|          | Mean      | MI-CAP (ave) | 0.196 ± 0.077 | 0.084 ± 0.055 | 0.060 ± 0.059 | 0.126 ± 0.023 | 0.084 ± 0.052 |
|          | Acc       | MI-CAP (ave) | 0.132 ± 0.084 | 0.176 ± 0.074 | 0.268 ± 0.027 | 0.128 ± 0.093 | 0.108 ± 0.063 |
|          | Rec       | MI-CAP (ave) | 0.196 ± 0.079 | 0.195 ± 0.062 | 0.140 ± 0.078 | 0.144 ± 0.039 | 0.112 ± 0.106 |
|          |           | MI-CAP (att) | 0.136 ± 0.086 | 0.132 ± 0.098 | 0.084 ± 0.094 | 0.080 ± 0.109 | 0.076 ± 0.108 |
|          |           | MI-CAP (att) | 0.244 ± 0.119 | 0.164 ± 0.071 | 0.092 ± 0.050 | 0.200 ± 0.071 | 0.212 ± 0.130 |
| STL10    | No        | MI-CAP (ave) | 0.804 ± 0.033 | 0.828 ± 0.054 | 0.820 ± 0.024 | 0.776 ± 0.003 | 0.808 ± 0.090 |
|          | Random    | MI-CAP (ave) | 0.112 ± 0.018 | 0.116 ± 0.070 | 0.096 ± 0.062 | 0.032 ± 0.043 | 0.108 ± 0.048 |
|          | Mean      | MI-CAP (ave) | 0.052 ± 0.076 | 0.196 ± 0.122 | 0.068 ± 0.038 | 0.124 ± 0.050 | 0.086 ± 0.058 |
|          | Acc       | MI-CAP (ave) | 0.176 ± 0.053 | 0.172 ± 0.172 | 0.192 ± 0.089 | 0.155 ± 0.062 | 0.228 ± 0.081 |
|          | Rec       | MI-CAP (ave) | 0.128 ± 0.076 | 0.240 ± 0.038 | 0.188 ± 0.057 | 0.252 ± 0.099 | 0.140 ± 0.073 |
|          |           | MI-CAP (att) | 0.124 ± 0.091 | 0.108 ± 0.035 | 0.072 ± 0.044 | 0.061 ± 0.052 | 0.132 ± 0.151 |
|          |           | MI-CAP (att) | 0.212 ± 0.104 | 0.176 ± 0.142 | 0.240 ± 0.069 | 0.036 ± 0.103 | 0.065 ± 0.087 |
|          | No        | DSMIL (ave)  | 0.756 ± 0.067 | 0.759 ± 0.049 | 0.542 ± 0.192 | 0.162 ± 0.072 | 0.205 ± 0.192 |
|          | Random    | DSMIL (ave)  | 0.251 ± 0.081 | 0.122 ± 0.135 | 0.137 ± 0.094 | 0.080 ± 0.130 | 0.080 ± 0.130 |
|          | Mean      | DSMIL (ave)  | 0.220 ± 0.173 | 0.289 ± 0.128 | 0.208 ± 0.128 | 0.087 ± 0.135 | 0.159 ± 0.135 |
|          | Acc       | DSMIL (ave)  | 0.240 ± 0.181 | 0.190 ± 0.087 | 0.135 ± 0.057 | 0.135 ± 0.057 | 0.135 ± 0.057 |
|          | Rec       | DSMIL (ave)  | 0.195 ± 0.057 | 0.320 ± 0.118 | 0.119 ± 0.156 | 0.119 ± 0.156 | 0.119 ± 0.156 |
|          |           | DSMIL (att)  | 0.251 ± 0.192 | 0.112 ± 0.070 | 0.115 ± 0.083 | 0.115 ± 0.083 | 0.115 ± 0.083 |
|          |           | DSMIL (att)  | 0.361 ± 0.077 | 0.304 ± 0.105 | 0.029 ± 0.085 | 0.029 ± 0.085 | 0.029 ± 0.085 |

Therefore, we show in this table the accuracy and standard deviation of the training data set. Here, we also did not conduct parameter analysis tests because $\xi = 0.01$ produced good results. The results reveal that VAD methods require more safety concerns since our techniques perform well on the VAD data sets.

4.4. Generalizability Comparison

We demonstrate the generalizability of the algorithm from two aspects, including applying perturbations generated on one network to other networks and to traditional MIL methods.

Figure 3 shows an example of the generalizability of MI-UAP on the MNIST data set. Each row depicts the result of attacking other algorithms with the perturbation generated by the current algorithm. The diagonal lines specifically refer to the classification results in Table 2. The hue of the square box gets darker as the value increases, signifying a stronger generalizability of the perturbation. The results demonstrate that perturbations to one algorithm can effectively attack others, possibly with higher fooling rates. This also illustrates the need for developing MI-UAP from another perspective, as it can generate only one perturbation for one data set, making it easy to store and use right away.

Specifically, we use the perturbations generated by MIL neural networks on the benchmark data sets to fool the traditional MIL methods, as shown in Table 3. The major goal of this experiment is to confirm that using MI-UAP to fool the traditional MIL ones is feasible because adding perturbations to drug molecules is unpractical. The results show that our method works in most cases, with limited underperformance on BAMIC and ELDB on some data sets. It seems like a coincidence that both algorithms are bag-based embedding methods. Therefore, we speculate that the possible reason is that the added perturbation occasionally does not alter the metric relationship based on bag-level distance.

4.5. Perturbation Defence

Now that we have designed two methods to expose the vulnerability of the MIL embedding methods, some algorithm protection tactics should be taken into account to ensure the practical use of the MIL methods. A simple and efficient strategy is
Table 2: Performance comparison for network-based classification on three databases using five perturbation strategies ($\xi = 0.01$).

| Data set       | Evaluator | Perturbation | ABMIL $[8]$ | GAMIL $[8]$ | LAMIL $[25]$ | DSMIL $[14]$ | MAMIL $[10]$ |
|---------------|-----------|--------------|-------------|-------------|--------------|--------------|--------------|
| ShanghaiTech  | Acc       | No           | 0.984 ± 0.05 | 0.979 ± 0.05 | 1.000 ± 0.00 | 1.000 ± 0.00 | 0.986 ± 0.016 |
|               |           | MI-CAP (ave) | 0.602 ± 0.026 | 0.633 ± 0.065 $\downarrow$ | 0.611 ± 0.028 $\downarrow$ | 0.435 ± 0.016 $\downarrow$ | 0.349 ± 0.019 $\downarrow$ |
|               |           | MI-CAP (att) | 0.621 ± 0.019 $\downarrow$ | 0.401 ± 0.013 $\downarrow$ | 0.607 ± 0.034 | 0.429 ± 0.013 $\downarrow$ | 0.355 ± 0.026 $\downarrow$ |
|               |           | MI-UAP (ave) | 0.402 ± 0.012 $\downarrow$ | 0.398 ± 0.019 | 0.449 ± 0.022 $\downarrow$ | 0.424 ± 0.017 | 0.378 ± 0.058 |
|               |           | MI-UAP (att) | 0.401 ± 0.013 $\downarrow$ | 0.439 ± 0.057 $\downarrow$ | 0.430 ± 0.025 $\downarrow$ | 0.414 ± 0.018 | 0.415 ± 0.005 $\downarrow$ |
|               | Rec       | No           | 0.940 ± 0.021 | 0.921 ± 0.019 | 1.000 ± 0.000 | 1.000 ± 0.000 | 0.946 ± 0.061 |
|               |           | MI-CAP (ave) | 0.307 ± 0.022 | 0.279 ± 0.031 | 0.320 ± 0.019 $\downarrow$ | 0.822 ± 0.031 $\downarrow$ | 0.660 ± 0.030 $\downarrow$ |
|               |           | MI-CAP (att) | 0.298 ± 0.016 $\downarrow$ | 0.752 ± 0.022 $\downarrow$ | 0.307 ± 0.036 $\downarrow$ | 0.810 ± 0.025 $\downarrow$ | 0.660 ± 0.022 $\downarrow$ |
|               |           | MI-UAP (ave) | 0.746 ± 0.017 $\downarrow$ | 0.721 ± 0.047 $\downarrow$ | 0.787 ± 0.027 $\downarrow$ | 0.800 ± 0.033 $\downarrow$ | 0.800 ± 0.094 $\downarrow$ |
|               |           | MI-UAP (att) | 0.651 ± 0.022 $\downarrow$ | 0.674 ± 0.113 $\downarrow$ | 0.794 ± 0.037 $\downarrow$ | 0.754 ± 0.023 $\downarrow$ | 0.790 ± 0.027 $\downarrow$ |
| UCF-Crime     | Acc       | No           | 0.991 ± 0.002 | 0.983 ± 0.007 $\downarrow$ | 0.997 ± 0.002 $\downarrow$ | 0.999 ± 0.001 $\downarrow$ | 0.991 ± 0.003 $\downarrow$ |
|               |           | MI-CAP (ave) | 0.498 ± 0.015 | 0.504 ± 0.005 $\downarrow$ | 0.510 ± 0.012 $\downarrow$ | 0.695 ± 0.277 $\downarrow$ | 0.488 ± 0.024 $\downarrow$ |
|               |           | MI-CAP (att) | 0.489 ± 0.023 | 0.470 ± 0.026 $\downarrow$ | 0.498 ± 0.008 $\downarrow$ | 0.685 ± 0.234 $\downarrow$ | 0.485 ± 0.012 $\downarrow$ |
|               |           | MI-UAP (ave) | 0.507 ± 0.014 $\downarrow$ | 0.495 ± 0.019 $\downarrow$ | 0.507 ± 0.007 $\downarrow$ | 0.527 ± 0.017 $\downarrow$ | 0.498 ± 0.005 $\downarrow$ |
|               |           | MI-UAP (att) | 0.501 ± 0.007 $\downarrow$ | 0.488 ± 0.003 $\downarrow$ | 0.515 ± 0.009 $\downarrow$ | 0.526 ± 0.022 $\downarrow$ | 0.504 ± 0.007 $\downarrow$ |

Figure 6: The generalizability experiments of MI-UAP are achieved by applying perturbations generated on one network to other networks ($\xi = 0.2$).

Table 3: Performance comparison for classification on five MIL benchmark data sets using MI-UAP (att) with ABMIL.

| Data set       | BAMIC $[11]$ | mFV $[31]$ | msVLAD $[31]$ | ELDB $[51]$ | AEMI $[37]$ |
|---------------|--------------|------------|---------------|-------------|-------------|
| Musk1         | 0.885 ± 0.034 $\rightarrow$ 0.860 ± 0.011 $\rightarrow$ 0.887 ± 0.025 $\rightarrow$ 0.484 ± 0.009 $\rightarrow$ 0.829 ± 0.050 $\rightarrow$ 0.664 ± 0.050 $\rightarrow$ 0.902 ± 0.016 $\rightarrow$ 0.853 ± 0.012 $\rightarrow$ 0.867 ± 0.019 $\rightarrow$ 0.517 ± 0.021 |
| Musk2         | 0.868 ± 0.022 $\rightarrow$ 0.774 ± 0.024 $\rightarrow$ 0.748 ± 0.034 $\rightarrow$ 0.622 ± 0.007 $\rightarrow$ 0.788 ± 0.041 $\rightarrow$ 0.642 ± 0.032 $\rightarrow$ 0.857 ± 0.039 $\rightarrow$ 0.798 ± 0.034 $\rightarrow$ 0.804 ± 0.010 $\rightarrow$ 0.384 ± 0.009 |
| Elephant      | 0.844 ± 0.013 $\rightarrow$ 0.769 ± 0.013 $\rightarrow$ 0.848 ± 0.008 $\rightarrow$ 0.580 ± 0.000 $\rightarrow$ 0.859 ± 0.031 $\rightarrow$ 0.420 ± 0.020 $\rightarrow$ 0.843 ± 0.012 $\rightarrow$ 0.757 ± 0.015 $\rightarrow$ 0.875 ± 0.010 $\rightarrow$ 0.461 ± 0.049 |
| Fox           | 0.647 ± 0.020 $\rightarrow$ 0.658 ± 0.004 $\rightarrow$ 0.639 ± 0.010 $\rightarrow$ 0.500 ± 0.000 $\rightarrow$ 0.611 ± 0.020 $\rightarrow$ 0.427 ± 0.027 $\rightarrow$ 0.648 ± 0.014 $\rightarrow$ 0.501 ± 0.030 $\rightarrow$ 0.583 ± 0.019 $\rightarrow$ 0.423 ± 0.022 |
| Tiger         | 0.797 ± 0.009 $\rightarrow$ 0.692 ± 0.008 $\rightarrow$ 0.646 ± 0.032 $\rightarrow$ 0.491 ± 0.021 $\rightarrow$ 0.701 ± 0.044 $\rightarrow$ 0.427 ± 0.037 $\rightarrow$ 0.767 ± 0.013 $\rightarrow$ 0.683 ± 0.012 $\rightarrow$ 0.795 ± 0.010 $\rightarrow$ 0.477 ± 0.045 |

To add a few adversarial examples to the training set, as shown in Figure 7, which shows the classification results of adversarial examples generated based on MI-CAP (att). The reason for choosing MI-CAP is that it is more aggressive and has a good ability to fool the classifier. The findings demonstrate that, although this is not as good as the performance without perturbation, only a small number of adversarial examples are needed to considerably increase the robustness of the classifier. Why not keep the percentage of adversarial examples rising? Because we believe that rather than doing something as straightforward as adding a lot of adversarial examples, our future focus should be on enhancing the safety of the algorithm itself.
5. Conclusions

In this paper, we have proposed two perturbation generation algorithms, MI-CAP and MI-UAP, to somewhat fool the MIL learner and interpret its vulnerability. Extensive experiments are conducted to demonstrate the effectiveness and generalizability of the proposed algorithms. A simple strategy is also proposed to deal with these adversarial perturbations. This helps us to understand more clearly about the factors that influence the security of an algorithm, and actually provides tactics for enhancing it. In future, we plan to focus more on developing algorithms that are more secure. Specifically, in view of developing a more secure MIL algorithm, the following points can be considered.

1. In the generated perturbations, more attention should be paid to the valuable parts of the bag, such as instances that can trigger the bag label and essential areas within instances.

2. Consider important details or unique circumstances in the actual application scenario to generate more targeted perturbation.

3. Attempts to identify perturbations in input data to rule out or lessen the impact of adversarial perturbations.

Declaration of Competing Interest

No conflict of interest exits in the submission of this manuscript, and manuscript is approved by both authors for publication. We declare that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (62131016) and Open Project of Zhejiang Key Laboratory of Marine Big Data Mining and Application (OBDMA202102).

References

[1] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In NeurIPS, pages 561–568, 2002.
[2] Gabriele Campanella, Matthew G Hanna, Luke Geneslaw, Allen Mirafior, Vitor Werneck-Krauss-Silva, Klaus J Busam, Edi Brogi, Victor E Reuter, David S Klimestone, and Thomas J Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. Nature Medicine, 25(8):1301–1309, 2019.
[3] Yi Xin Chen, Jin Bo Bi, and J. Z. Wang. MILES: Multiple-instance learning via embedded instance selection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12):1931–1947, 2006.
[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In AISTATS, pages 215–223, 2011.
[5] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence, 89:31–71, 1997.
[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv, pages 1–11, 2014.
[7] Shi Luo Huang, Zheng Liu, Wei Jin, and Ying Mu. Bag dissimilarity regularized multi-instance learning. Pattern Recognition, 126:108583, 2022.
[8] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In ICML, pages 2127–2136, 2018.
[9] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Hillier Chloe, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. arXiv, pages 1–22, 2017.
[10] Andrei V Konstantinov and Lev V Utkin. Multi-attention multiple instance learning. Neural Computing and Applications, pages 1–23, 2022.
