Generating Multi-label Adversarial Examples by Linear Programming

Nan Zhou*, Wenjian Luo†, Xin Lin*, Peilan Xu* and Zhenya Zhang‡
*School of Computer Science and Technology, University of Science and Technology of China
Hefei 230027, Anhui, China
{hinanmu, iskcal, xpl}@mail.ustc.edu.cn
†School of Computer Science and Technology, Harbin Institute of Technology
Shenzhen 518055, Guangdong, China
luowenjian@hit.edu.cn
‡Anhui Province Key Laboratory of Intelligent Building and Building Energy Saving, Anhui Jianzhu University
Hefei 230022, Anhui, China
zyzhang@ahjzu.edu.cn

***Abstract*—Deep neural networks (DNNs) are used in various domains, such as image classification, natural language processing and face recognition, etc. However, the presence of malicious examples, generated by specific methods, could result in DNNs misclassification. Such maliciously modified examples are called adversarial examples. So far, most work about adversarial examples mainly focuses on the multi-class classification tasks, and only a little work has been done in the field of multi-label classification.**

In this study, we have proposed a novel algorithm that generates effective multi-label adversarial examples by solving a linear programming problem (MLA-LP). We minimize the l_∞ norm of distortion while constraining the changes in the label loss of the example after being perturbed. Then, we transform this constrained optimization problem into a linear programming problem for reducing the time cost. In comparison to the existing multi-label classification model attack algorithms, the attack performance of the proposed MLA-LP is found to be competitive, and the adversarial examples generated by MLA-LP have significantly smaller distortions.

***Index Terms*—Deep Neural Networks, Multi-label Classification, Adversarial Examples, Linear Programming**

I. Introduction

Deep neural networks (DNNs) have been playing an increasingly important role in the fields of image recognition, speech recognition, and natural language processing [1]. However, in recent years, security issues related to DNNs have been significantly emphasized [2]. When a small distortion, which cannot be recognized by humans, is added to a well-classified image, the confidence level of the wrong category can be considerably increased, thereby resulting in the classifier misclassification. These findings have made it difficult to guarantee security in some deep learning application systems, such as face recognition and autonomous driving. Typically, Szegedy *et al.* [2] defined the examples that are originally classified correctly after being perturbed to cause model misclassification as adversarial examples. Adversarial examples have become one of the main risks that threaten the security of deep learning models.

Existing work about generating adversarial examples is mainly dedicated to the field of multi-class classification, wherein the classification label has only one positive label. Accordingly, when the confidence level of any one of the negative labels becomes higher than that of the only one positive label, the attack is successful [3].

However, in the real world, many objects contain multiple labels. For example, a landscape image contains mountains, water, and clouds, while a piece of text may contain multiple categories of emotions. Correspondingly, multi-label adversarial examples are more likely to exist in real-world applications and they pose a threat to our daily lives. For example, autonomous driving vision systems can recognize multiple objects simultaneously, such as speed limit signs, vehicles, and pedestrians. If an adversary successfully hides any one of the positive labels, while the classifier fails to predict this label, the system will encounter serious consequences.

In DNNs for multi-label classification, if an adversary attempts to attack one or more specific labels, it is likely to change the confidence of the remaining non-attack labels. This is because the multi-label predictors depend on a specified threshold and the labels are often correlated [4]. Thus, the attack techniques for multi-class classification cannot be directly applied to DNNs for multi-label classification.

To the best of our knowledge, only [4], [5] have focused on multi-label adversarial examples. Song *et al.* [4] proposed a multi-label attack framework based on classification and ranking, and presented some multi-label attack methods based on existing multi-class attack techniques. Some methods achieve a high attack success rate, but the distortions in the adversarial examples are relatively large. Wu *et al.* [5] adopted multi-label adversarial examples for adversarial training to improve the robustness of the models.

In this study, the problem of generating multi-label adversarial examples is transformed into a linear programming problem, which minimizes the maximum distortions added

This study is partially supported by the open project of the Anhui Province Key Laboratory of Intelligent Building and Building Energy Saving (No. IBBE2018KX01ZD). *(Corresponding author: Wenjian Luo.)*

over all dimensions (l_∞ norm). By solving this linear programming problem, we can obtain a small distortion in each dimension; meanwhile, the linear problem-solving speed is very fast. Thus, the proposed algorithm is named MLA-LP. Experimental results for two multi-label classification models and two datasets indicate that the performance of MLA-LP is very comparable.

The rest of this paper is organized as follows. Section II introduces some background details, including multi-label classification and adversarial attacks. The proposed method is described in section III. Section IV discusses the experimental setting, results, and analyses. Section V presents a brief summary of this work.

II. Related Work

A. Notation Definition

Multi-label classification can be described as the d-dimensional input space $\mathcal{X} \in \mathbb{R}^d$ is mapped to the l-dimensional output space $\mathcal{Y} \in \{-1, 1\}^l$. That is, for each input vector $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$, the corresponding output $\mathbf{y} = [y_1, y_2, \cdots, y_l]^T$, where $y_j = 1$ ($j = 1, 2, ..., l$), which implies that $\mathbf{x}$ owns the label y_j; otherwise, $y_j = -1$.

Generally, in the prediction phase, for an example $\mathbf{x}$, the output confidence $\mathbf{o} = [o_1, o_2, \cdots, o_l]^T$ for a series of labels is obtained through a map function $f(\cdot)$, and the prediction of the example is $\mathbf{y}' = H(f(\mathbf{x}))$. Here, $H(\cdot)$ is a function indicating that the label is positive when the output is greater than or equal to the threshold t, while it is negative when the output is less than t, as shown below.

$$y_j^{'} = \begin{cases} 1, & \text{if } o_j \geq t \\ -1, & \text{if } o_j < t \end{cases} \tag{1}$$

Without loss of generality, the problem of generating adversarial examples can be considered as a constrained optimization problem. The objective is to minimize the amount of distortion while constraining the example after adding the distortion $\mathbf{r}$ to be classified as target labels $\mathbf{y}^*$, which is defined as [4]

$$\begin{aligned} \min_{\mathbf{r}} \quad & ||\mathbf{r}|| \\ \text{s.t.} \quad & H(f(\mathbf{x}^*)) = \mathbf{y}^*. \end{aligned} \tag{2}$$

Here, the distortion is denoted as $\mathbf{r} = [r_1, r_2, ..., r_d]^T$, $||\cdot||$ denotes norm term of $\mathbf{r}$, d is the dimension size of the input space, $\mathbf{x}^* = \mathbf{x} + \mathbf{r}$, and $\mathbf{y}^*$ is the expected label vector of the adversarial example. If the ground-truth label vector of example $\mathbf{x}$ is $\mathbf{y}$, $\mathbf{y}^* \neq \mathbf{y}$.

B. Multi-label Classification

Multi-label classification has been studied for many years and many algorithms have been proposed for it, including binary relevance [6], MLKNN [7], along with classifier chain [8], and its variants [9]–[11].

Neural networks are a popular solution to multi-label classification problems [12]–[14]. Back-propagation for multi-label learning (BPMLL) [15] is a typical multi-label classification algorithm based on neural networks. It exploits the loss function to mine label dependencies. Many algorithms are extensions or improvements of BPMLL. For example, Rafał *et al.* [16] modified the loss function of BPMLL.

CNN-WARP [17] extends a convolutional neural network (CNN) to the multi-label image classification field, which modifies the loss function to make the algorithm more suitable for multi-label image classification. Based on CNNs, CNN-RNN [18] attempts to encode the label information with the long short-term memory (LSTM) [19] to obtain the dependencies between labels. ML-GCN [20] encodes the dependencies between labels by constructing a directed acyclic graph (DAG) between the labels and processing the label graph using a graph convolutional network (GCN) [21].

C. Adversarial Attack

DNNs have achieved excellent results in many complex tasks, such as image classification. However, adversarial attacks could damage the networks if tiny noise is added to the input to cause model misclassification. Recently, an increasing number of studies has indicated that the security of DNNs is a problem that cannot be ignored.

1) Multi-class Adversarial Attack: Goodfellow *et al.* [3] believed that the adversarial examples are caused by the linearity of the deep learning model, and proposed a method (named FGSM) for quickly generating adversarial examples based on the gradient sign. There are many variants based on FGSM [22]–[24]. Nicolas *et al.* [25] used the Jacobian matrix to determine the pixels that can reduce the confidence of the correct category and increase the confidence of the wrong category, and iteratively modified these pixels to achieve the goal of attacking the model. Luo *et al.* [26] proposed an algorithm named RDA, which implements the hill-climbing method to determine a direction that can increase the confidence of the wrong category. The above algorithms are mainly verified in the field of image classification. At present, there are also some studies in the field of speech and text classification [27]–[29].

2) Multi-label Adversarial Attack: Unlike the multi-class model, which only outputs one positive label, the multi-label model outputs multiple positive labels. The multi-label classification model often sets a threshold for the confidence of classification. If the confidence is greater than or equal to the threshold, the label is positive. Otherwise, the label is negative. When attacking a multi-label DNN model, we want to make the output probability changes of the attacked labels as large as possible and ensure that the confidence changes of other non-attacked labels within a reasonable range. This can make these non-attacked labels still be classified correctly after the attack. Thus, multi-class adversarial attack methods cannot be directly applied to the multi-label model.

As stated in Section I, there are only two studies that focus on multi-label adversarial attacks. In [5], Wu *et al.* used adversarial training to improve the performance of specific algorithms in text relation extraction tasks, where multi-label adversarial examples were used to train the model. In [4], based on the existing popular multi-class model attack methods C&W [30], DeepFool [31], and FGSM [3], Song *et* *al.* proposed multiple classification and ranking-based multi-label adversarial attack methods, such as ML-DeepFool and ML-CW.

ML-DeepFool proposed in [4], which is based on DeepFool [31], is closely related to this study, and it will be introduced in the next subsection.

D. ML-DeepFool

ML-DeepFool [4] was inspired by DeepFool [31]. It is assumed that the influence of the distortion added to the example on the output of the model is linear. By calculating the gradient of the output to the input as the weight of the distortion on output change, the corresponding amount of distortion can be calculated by defining a desired output according to the attack goal.

The optimization goal of the algorithm in [4] is defined as

$$\begin{aligned} \min_{\mathbf{r}} \quad & ||\mathbf{r}||_2 \\ \text{s.t.} \quad & -\mathbf{y}^* \odot f(\mathbf{x}^*) \leq -\mathbf{y}^* \odot \mathbf{t}, \end{aligned} \tag{3}$$

where $\mathbf{r} \in \mathbb{R}^d$ denotes the distortion and d denotes the dimension of the input vector. $||\cdot||_2$ represents the l_2 norm, $\odot$ represents the Hadamard product, $\mathbf{x}^* = \mathbf{x} + \mathbf{r}$, $f(\cdot) \in \mathbb{R}^l$ is a multi-label classification model, l is the dimension of the label, $\mathbf{t} = [\underbrace{t, t, \cdots, t}_{l}]^T$ is the threshold constant vector, and t is set to 0.5 when the model output $f(\cdot)$ is in the interval $[0, 1]$. The symbol '$\leq$' between the vectors means that the value of each dimension of the left vector must be no greater than the value of the corresponding dimension of the right vector.

Because the ML-DeepFool algorithm assumes that the effect of distortion on the model output is linear, formula (3) can be approximately transformed to formula (4).

$$\begin{aligned} \min_{\mathbf{r}} \quad & ||\mathbf{r}||_2 \\ \text{s.t.} \quad & -\mathbf{y}^* \odot (f(\mathbf{x}) + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{r}) \leq -\mathbf{y}^* \odot \mathbf{t}. \end{aligned} \tag{4}$$

For simplification, ML-DeepFool does not solve the objective function $\min_{\mathbf{r}} \|\mathbf{r}\|_2$. Instead, it directly transforms the inequality constraint into an equation, and solves this equation to obtain the solution $\mathbf{r}$, which is shown as follows.

$$\begin{aligned} & -\mathbf{y}^* \odot (f(\mathbf{x}) + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\mathbf{r}) = -\mathbf{y}^* \odot \mathbf{t} \\ & \mathbf{r} = (\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}})((\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}})^T(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}))^{-1}(\mathbf{t} - f(\mathbf{x})). \end{aligned} \tag{5}$$

The distortion $\mathbf{r}$ is calculated by the above formula. Then, the adversarial example is $\mathbf{x}^* = \mathbf{x} + \mathbf{r}$. ML-DeepFool applies the above method iteratively and continuously updates $\mathbf{x}^*$ until the attack succeeds or the maximum number of iterations is reached.

III. Proposed Method

A. Basic Idea

ML-DeepFool defines the problem of generating adversarial example as a constrained optimization problem, as shown in (4), which includes constraining the output of the example after adding the distortion while minimizing the l_2 norm of the distortion amount. However, as shown in formula (5), ML-DeepFool does not solve this constrained optimization problem directly but uses a greedy method to solve its constraint condition. This often leads to a larger distortion in generating an effective adversarial example.

In this study, to obtain smaller distortion and generate an adversarial example quickly, we define the objective function of the constrained optimization problem to minimize the l_∞ norm of the distortion amount; then, it is transformed into a linear programming problem. Thus, the well-studied linear programming methods can be used to solve this problem.

B. MLA-LP

Because the DNN models are trained by minimizing the loss function, there is a tight correlation between the prediction of the model and the loss. For convenience, we define the target loss vector as

$$loss(\mathbf{t}, \mathbf{y}^*) = [loss_1, loss_2, \cdots, loss_l]^T,$$

where $\mathbf{y}^*$ is the prediction target, and $loss_i \in loss(\mathbf{t}, \mathbf{y}^*)$ is the specific loss for the threshold $t_i \in \mathbf{t}$.

We use the Jacobian matrix to measure the effect of changes in input on the labels. The output of the model is not used directly to get the Jacobian matrix like ML-DeepFool [4]. Instead, MLA-LP calculates the first-order partial derivative of the loss of the output against the input to get the Jacobian matrix. It is believed that the changes in the input will be more sensitive to those in the loss. The matrix is defined as

$$\begin{aligned} \mathbf{A} &= \frac{\partial loss(f(\mathbf{x}), \mathbf{y}^*)}{\partial \mathbf{x}} \\ &= \begin{bmatrix} \frac{\partial loss(f_1(\mathbf{x}),y_1^*)}{\partial x_1} & \frac{\partial loss(f_1(\mathbf{x}),y_1^*)}{\partial x_2} & \cdots & \frac{\partial loss(f_1(\mathbf{x}),y_1^*)}{\partial x_d} \\ \frac{\partial loss(f_2(\mathbf{x}),y_2^*)}{\partial x_1} & \frac{\partial loss(f_2(\mathbf{x}),y_2^*)}{\partial x_2} & \cdots & \frac{\partial loss(f_2(\mathbf{x}),y_2^*)}{\partial x_d} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial loss(f_l(\mathbf{x}),y_l^*)}{\partial x_1} & \frac{\partial loss(f_l(\mathbf{x}),y_l^*)}{\partial x_2} & \cdots & \frac{\partial loss(f_l(\mathbf{x}),y_l^*)}{\partial x_d} \end{bmatrix}, \end{aligned} \tag{6}$$

where $\mathbf{A}$ is the Jacobian matrix and $loss(\cdot)$ is a specific loss function. In the case of very small distortion, the changes in loss and distortion are approximately linear. That is,

$$loss(f(\mathbf{x} + \mathbf{r}), \mathbf{y}^*) \approx loss(f(\mathbf{x}), \mathbf{y}^*) + \frac{\partial loss(f(\mathbf{x}), \mathbf{y}^*)}{\partial \mathbf{x}} \cdot \mathbf{r}. \tag{7}$$

Owing to the classification threshold in multi-label classification models, it is necessary to significantly change the confidence of attacked labels while ensuring that the confidence changes in other non-attacked labels are minimal. Therefore, we make the loss in the model output of the adversarial example is less than the target loss $loss(\mathbf{t}, \mathbf{y}^*)$, and this will enable the confidence of the attacked labels to cross the threshold and prevent the non-attacked label from crossing the threshold. That is,

$$loss(f(\mathbf{x} + \mathbf{r}), \mathbf{y}^*) \leq loss(\mathbf{t}, \mathbf{y}^*), \tag{8}$$

where '$\leq$' between the vectors indicates that the value of each dimension of the left vector must be less than the value of the corresponding dimension of the right vector.

Now, we substitute formula (7) into formula (8), and define the optimization problem as

$$\begin{aligned} \min_{\mathbf{r}} \quad & ||\mathbf{r}||_{\alpha} \\ \text{s.t.} \quad & \frac{\partial loss(f(\mathbf{x}),\mathbf{y}^*)}{\partial \mathbf{x}} \cdot \mathbf{r} \le loss(\mathbf{t},\mathbf{y}^*) - loss(f(\mathbf{x}),\mathbf{y}^*). \end{aligned} \tag{9}$$

Here, α could denote any norm.

Evidently, if α=2, formula (9) is similar to formula (4), except that it uses the loss function. To effectively solve the above constraint optimization problem, the norm term of the objective function is selected to use l_∞ as the minimization goal. Thus, we have

$$\begin{aligned} \min_{\mathbf{r}} \quad & ||\mathbf{r}||_{\infty} \\ \text{s.t.} \quad & \frac{\partial loss(f(\mathbf{x}),\mathbf{y}^*)}{\partial \mathbf{x}} \cdot \mathbf{r} \le loss(\mathbf{t},\mathbf{y}^*) - loss(f(\mathbf{x}),\mathbf{y}^*), \end{aligned} \tag{10}$$

where l_∞ of the distortion amount is the maximum absolute value of the distortion amount, i.e., $\|\mathbf{r}\|_\infty = max\{|r_i| : i = 1, 2, \ldots, n\}$. Now, our objective function becomes

$$\begin{aligned} \min_{\mathbf{r}} \quad & max\{|r_i| : i = 1, 2, \ldots, d\} \\ \text{s.t.} \quad & \frac{\partial loss(f(\mathbf{x}),\mathbf{y}^*)}{\partial \mathbf{x}} \cdot \mathbf{r} \le loss(\mathbf{t},\mathbf{y}^*) - loss(f(\mathbf{x}),\mathbf{y}^*). \end{aligned} \tag{11}$$

To effectively solve the above optimization problem with the non-linear objective function, we need to convert this objective function into a linear objective function. Thus, we introduce a variable z and define

$$z \ge |r_i|, \quad i = 1, 2, ..., d. \tag{12}$$

It should be noted that formula (12) can be transformed into

$$\begin{aligned} & z \ge -r_i, \ i = 1, 2, ..., d, \\ & z \ge r_i, \quad i = 1, 2, ..., d. \end{aligned} \tag{13}$$

Finally, the optimization problem (11) is transformed into the following formula.

$$\begin{aligned} \min_{z,\mathbf{r}} \quad & z \\ \text{s.t.} \quad & z \ge r_i, \quad i = 1, 2, ..., d, \\ & z \ge -r_i, \ i = 1, 2, ..., d, \\ \frac{\partial loss(f(\mathbf{x}),\mathbf{y}^*)}{\partial \mathbf{x}} & \cdot \mathbf{r} \le loss(\mathbf{t},\mathbf{y}^*) - loss(f(\mathbf{x}),\mathbf{y}^*). \end{aligned} \tag{14}$$

By solving the above linear programming problem, we can get z and $\mathbf{r}$; consequently, the adversarial example is

$$\mathbf{x}^* = \mathbf{x} + \mathbf{r}. \tag{15}$$

Owing to the use of linear hypothesis, the adversarial example obtained by one iteration may not perform successful attack. Therefore, similar to ML-DeepFool [4] and DeepFool [31], we use multiple iterations to obtain the adversarial example. The algorithm flow of MLA-LP is presented in Algorithm 1.

In addition, for the linear programming problem shown in (14), because the dimension of the input example is too high, the coefficient matrix is too large to be stored directly. Therefore, we use the sparse matrix form and it only store the non-zero values in the original coefficient matrix, which enable us to solve it using the interior point method in the mosek package [32].

Algorithm 1 MLA-LP

Input: Image $\mathbf{x}$, ground-truth label $\mathbf{y}$, target label $\mathbf{y}^*$, map function f, classification function H, threshold vector $\mathbf{t}$, maximum iterations $maxiter$.

Output: Adversarial example $\mathbf{x}^*$.

```
1: Let $i = 0$, $\mathbf{y}' = \mathbf{y}$, $\mathbf{x}^* = \mathbf{x}$, $\mathbf{r}_0 = \mathbf{0}$.
2: while $i \le maxiter$ and $\mathbf{y}' \ne \mathbf{y}^*$ do
3:     $\mathbf{o} = f(\mathbf{x}^*)$
4:     $\mathbf{loss1} = loss(\mathbf{o}, \mathbf{y}^*)$
5:     $\mathbf{loss2} = loss(\mathbf{t}, \mathbf{y}^*)$
6:     $\mathbf{b} = \mathbf{loss2} - \mathbf{loss1}$
7:     $\mathbf{A} = \bigtriangledown_{\mathbf{x}^*}\mathbf{loss1}$
8:     Transform $\mathbf{A}$ and $\mathbf{b}$ to $\mathbf{A_z}$ and $\mathbf{b_z}$ according to (14)
9:     $(z, \mathbf{r}_{i+1}) = InteriorPointSolver(\mathbf{A_z}, \mathbf{b_z})$
10:    $\mathbf{x}^* = \mathbf{x}^* + \mathbf{r}_{i+1}$
11:    $\mathbf{y}' = H(f(\mathbf{x}^*))$
12: end while
13: return $\mathbf{x}^*$
```

IV. Experiments

A. Multi-label Classification Model

Two multi-label classification models are used in experiments, i.e., ML-GCN [20] and the model used in ML-DeepFool [4] for experimental verification. For convenience, the second one is named ML-LIW in this paper. The classification performance of ML-GCN is better than ML-LIW.

1) ML-GCN: ML-GCN uses a conditional probability between two labels in the dataset to obtain a directed acyclic graph (DAG). Then, it takes all labels' word vector as the node information of the DAG. It encodes this graph using a GCN [21] to obtain the correlations within the labels. The confidence of the label output is obtained by integrating the extracted feature information and the label information.

First, the image passes the CNN model to obtain a feature vector

$$\mathbf{feature} = f_{cnn}(\mathbf{x}), \tag{16}$$

where $\mathbf{feature} \in R^{d'}$, d' is the dimension of the fully connected layer of the CNN and the node feature dimension output in the last GCN layer.

Second, a GCN layer is expressed as

$$H_{out} = h(\hat{A}H_{in}W), \tag{17}$$

where $\hat{A} \in \mathbb{R}^{l*l}$ is the weight matrix of the DAG (the weight matrix is the conditional probability between all labels in the dataset). $H_{in} \in \mathbb{R}^{l*d}$ is a GCN input, d is the input feature dimension of each node, In the first GCN layer, H_{in} is initialized by the label's word vector matrix. $H_{out} \in \mathbb{R}^{l*d'}$ is the output node feature matrix, d' is the output feature dimension of each node, $h(\cdot)$ is a non-linear function, d is the input feature dimension of each node, and $W \in \mathbb{R}^{d*d'}$ is the model parameter that we need to train.

The output of the final model is

$$\mathbf{o} = \sigma(H_{out} * \mathbf{feature}), \tag{18}$$

where σ is the $sigmoid$ activation function.

2) ML-LIW: ML-LIW was used as the classification model in [20]. Based on BPMLL's [15] label-wise ranking loss, the instance-wise ranking loss is added. For the training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, 2, ..., n\}$, the loss function is

$$\begin{aligned} E =& \lambda \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathbf{y}_i||\overline{\mathbf{y}}_i|} \sum_{(k,l) \in \mathbf{y}_i \odot \overline{\mathbf{y}}_i} \exp\left(-\left(o_k^i - o_l^i\right)\right) \\ &+ \frac{1}{l} \sum_{j=1}^{l} \frac{1}{|\mathbf{y}_j||\overline{\mathbf{y}}_j|} \sum_{(p,q) \in \mathbf{y}_j \odot \overline{\mathbf{y}}_j} \exp\left(-\left(o_p^j - o_q^j\right)\right), \end{aligned} \tag{19}$$

where λ balances two types of losses, $\mathbf{y}_i$ represents the collection of labels belonging to the example $\mathbf{x}_i$, $\overline{\mathbf{y}}_i$ represents the collection of labels that are not a part of the example, and $|\cdot|$ represents the size of the collection. o_k^i and o_l^i indicate the neural network outputs corresponding to the labels $y_k^i \in \mathbf{y}_i$ and $y_l^i \in \overline{\mathbf{y}}_i$, respectively, where $\mathbf{y}_j$ represents the collection of instances that contain the j-th label, $\overline{\mathbf{y}}_j$ represents the collection of instances that don't have the j-th label, and o_p^j and o_q^j indicate the neural network outputs corresponding to the labels $y_p^j \in \mathbf{y}_j$ and $y_q^j \in \overline{\mathbf{y}}_j$, respectively.

B. Dataset

We chose two benchmark datasets, PASCAL VOC2007 [33] and VOC2012 [34], for multi-label image classification in the experiments.

We apply the classification algorithms ML-GCN and ML-LIW on VOC2007 and VOC2012, respectively. The size of all input images for ML-GCN is adjusted to 448×448 and that for ML-LIW is adjusted to 299×299; the image pixel values are normalized to [0, 1].

C. Model Training

The dataset VOC2007 contains a training set, a validation set, and a test set. However, the dataset VOC2012 only comprises a training set and a validation set. Therefore, for VOC2012, we randomly select 80% of the original training set as the real training set of the model, while the remaining 20% of the original training set is used as the real validation set. The original validation set is used as the test set.

For the ML-GCN model, the training parameters are consistent with those presented in [20], i.e., there are 100 training epochs; the learning rate is 0.01, initially, and it reduces by 10% every 40 epochs. Additionally, SGD is the optimizer used here. For the ML-LIW model, there are 100 training epochs, the learning rate is 0.001, and the Adam optimizer is used.

The evaluation indicators of the model include the hamming loss, ranking loss, micro-f1, macro-f1, and average precision [35]. The performances of the multi-label classification models on these two datasets are presented in Table I .

First, we use the trained model to make predictions for the examples in the test set and select all examples whose positive label number is greater than two as those to be attacked. The number of test examples and that of examples to be attacked are listed in Table II.

D. Compared Algorithms

We select four multi-label attack methods, i.e., ML-CW, ML-DeepFool, ML-Rank1, and ML-Rank2 [4] for comparisons. The experimental parameters of the algorithms to be compared are consistent with those presented in their original work. The maximum number of iterations for ML-CW, ML-Rank1, and ML-Rank2 is 1000, the number of binary searches is 10, and the initial value of λ is set to 1e5. The maximum number of iterations for ML-DeepFool is 20.

Additionally, the maximum number of iterations for MLA-LP is 10.

E. Experimental Results

In the experiments, we randomly select one label from the positive labels as the attack label. That is, we hide one label and construct the targeted attack label dataset accordingly.

The experimental results are presented in Table III, which shows the performance of all algorithms under the VOC2007 and VOC2012 datasets. Bold text indicates the best performance out of the five algorithms. The upward arrow in the table column name indicates that the larger the value, the better the algorithm's performance in the indicator; otherwise, the smaller the value, the better. The attack success rate in the table is the ratio of the number of examples of successful attacks to that of examples to be attacked. l_2, l_∞, and root mean squared deviation (RMSD) calculates the average of all successful adversarial example distortions, where l_2 norm and l_∞ norm are calculated by pixel values in [0, 1] and RMSD is calculated by pixel values in [0, 255], which is presented in [36].

From the table III, it is evident that in comparison to ML-DeepFool, MLA-LP achieves better performance for all indicators because ML-DeepFool does not fully consider the objective function (minimize the l_2 norm of distortion); it uses a greedy solution instead. Therefore, the distortions in the adversarial examples generated by it are very large, and the l_∞ norm is close to 1, which means that the pixels are flipped. A large distortion may reduce the confidence of the attacked label, but it also may change other non-attacked labels significantly, resulting in a lower attack success rate.

In comparison to other algorithms, the distortions observed in the adversarial examples of MLA-LP are relatively small in terms of the l_2 norm, l_∞ norm, and RMSD indicators. The attack success rate of MLA-LP is considerably higher than those of ML-Rank1 and ML-DeepFool, but lower than those of ML-CW and ML-Rank2. Because MLA-LP minimizes the amount of distortion in the l_∞ norm, we can obtain relatively

TABLE I
PERFORMANCE OF ML-GCN AND ML-LIW ON VOC2007 AND VOC2012 DATASETS

model	dataset	hamming loss	ranking loss	micro-f1	macro-f1	average precision
ML-GCN	VOC2007	0.0157	0.0065	0.8912	0.8747	0.9687
	VOC2012	0.0192	0.0098	0.8630	0.8512	0.9585
ML-LIW	VOC2007	0.0429	0.0219	0.7435	0.7560	0.9126
	VOC2012	0.0467	0.0212	0.7305	0.7443	0.9201

TABLE II
NUMBER OF TEST EXAMPLES AND EXAMPLES TO BE ATTACKED BY ML-GCN AND ML-LIW ON VOC2007 AND VOC2012 DATASETS

model	dataset	test examples	examples to be attacked
ML-GCN	VOC2007	4952	1113
	VOC2012	5823	862
ML-LIW	VOC2007	4952	739
	VOC2012	5823	585

small changes in each dimension, thereby resulting in a smaller l_∞ norm; meanwhile, the l_2 norm and RMSD also decrease.

It is also evident that the attack success rates of all attack methods over the ML-LIW classification model are higher than that of the ML-GCN classification model. This may due to the ML-GCN classification model is more complicated. The ML-GCN model uses a GCN to encode the correlations between labels based on Resnet101 [37]. Therefore, when an attack is performed, changes in the confidence of the attacked label will make the confidence of other related non-attacked labels more likely to change, thereby complicating the attack. The ML-LIW model adds a specific loss function based on inceptionV3 [38]. The correlation of the label is weak and the model is simpler than ML-GCN; thus, the attack success rate is higher.

F. An Example

To better understand the characteristics of MLA-LP, adversarial examples of an image in the dataset VOC2012 are shown in Fig. 1. The ML-GCN model is used in this experiment. Fig. 1 demonstrates the original example, the distortions in the three channels, and the adversarial examples for different attack methods.

From Fig. 1, it is evident that the original example contains the labels "Person" and "Tvmonitor". However, the adversarial examples by all attack methods are labeled with "Tvmonitor," but without "Person."

On observing the distortions of different channels, it is evident that the distortions produced by ML-DeepFool significantly vary and the values of distortions over many pixels are close to 1, which means that the pixel values are opposite. The distortions generated by the ML-CW algorithm are concentrated in some areas, the size of pixels in these areas varies between -0.01 and 0.01, and the sizes of remaining pixels are close to 0. The ML-Rank1 and ML-Rank2 algorithms interfere with almost all areas of the image, and the ML-Rank1 algorithm produces larger distortions. However, MLA-LP has smaller distortion, with values ranging between -0.005 and 0.005; it is also more uniform.

Overall, in all adversarial examples, one of them generated by ML-DeepFool is evident and it completely change the original image. The distortion example of ML-Rank1 is small; however, some of the noise observed in the image can be detected by the human eyes. The distortions generated by other algorithms are relatively small and these adversarial examples are very similar to the original example. The distortion amount in the adversarial example by MLA-LP is found to be the smallest.

V. CONCLUSION

DNNs have been widely implemented to solve multi-label classification problems. However, the security of DNNs for multi-label classification remains to be an issue. In this study, an algorithm to generate multi-label adversarial examples based on linear programming is proposed. This algorithm solves the distortion vector by constraining the label loss of the perturbed example while minimizing the l_∞ norm of distortion. Experimental results for the two classification models ML-GCN and ML-LIW and two datasets VOC2007 and VOC2012 demonstrate that the multi-label adversarial examples generated by the proposed algorithm have smaller distortions. In the future, we will attempt to verify the proposed method on more complex datasets.

ACKNOWLEDGMENT

The authors would like to thank Chen *et al.* [20] for providing the code of ML-GCN (https://github.com/Megvii-Nanjing/ML-GCN). The source code of this work is available at https://github.com/hinanmu/MLALP.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[4] Q. Song, H. Jin, X. Huang, and X. Hu, "Multi-label adversarial perturbations," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1242–1247.
[5] Y. Wu, D. Bamman, and S. Russell, "Adversarial training for relation extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1778–1783.
[6] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

TABLE III
EXPERIMENTAL RESULTS FOR VOC2007 AND VOC2012 DATASETS

model	dataset	attack method	attack success rate(↑)	RMSD(↓)	l_2 norm(↓)	l_∞ norm(↓)
ML-GCN	VOC2007	ML-CW	**0.9757**	0.2868	1.7456	0.0243
		ML-Rank1	0.1931	1.1466	6.9787	0.0357
		ML-Rank2	0.9128	1.1637	7.0824	0.0395
		ML-DeepFool	0.0000	-	-	-
		MLA-LP	0.8751	**0.0917**	**0.5585**	**0.0009**
	VOC2012	ML-CW	**0.9361**	0.3229	1.9652	0.0151
		ML-Rank1	0.2227	1.2951	7.8825	0.0318
		ML-Rank2	0.8863	1.1687	7.1130	0.0323
		ML-DeepFool	0.0092	73.6378	448.1560	1.0000
		MLA-LP	0.8329	**0.0399**	**0.2432**	**0.0004**
ML-LIW	VOC2007	ML-CW	**0.9972**	0.3204	1.3014	0.0423
		ML-Rank1	0.2070	1.8699	7.5953	0.0496
		ML-Rank2	0.9296	2.0951	8.5102	0.0717
		ML-DeepFool	0.4492	64.0180	260.0303	0.9978
		MLA-LP	0.9052	**0.2418**	**0.9824**	**0.0026**
	VOC2012	ML-CW	**0.9948**	0.3209	1.3036	0.0349
		ML-Rank1	0.1487	1.8550	7.5349	0.0454
		ML-Rank2	0.8632	1.8746	7.6146	0.0578
		ML-DeepFool	0.3914	67.4167	273.8353	0.9962
		MLA-LP	0.8888	**0.1952**	**0.7928**	**0.0020**

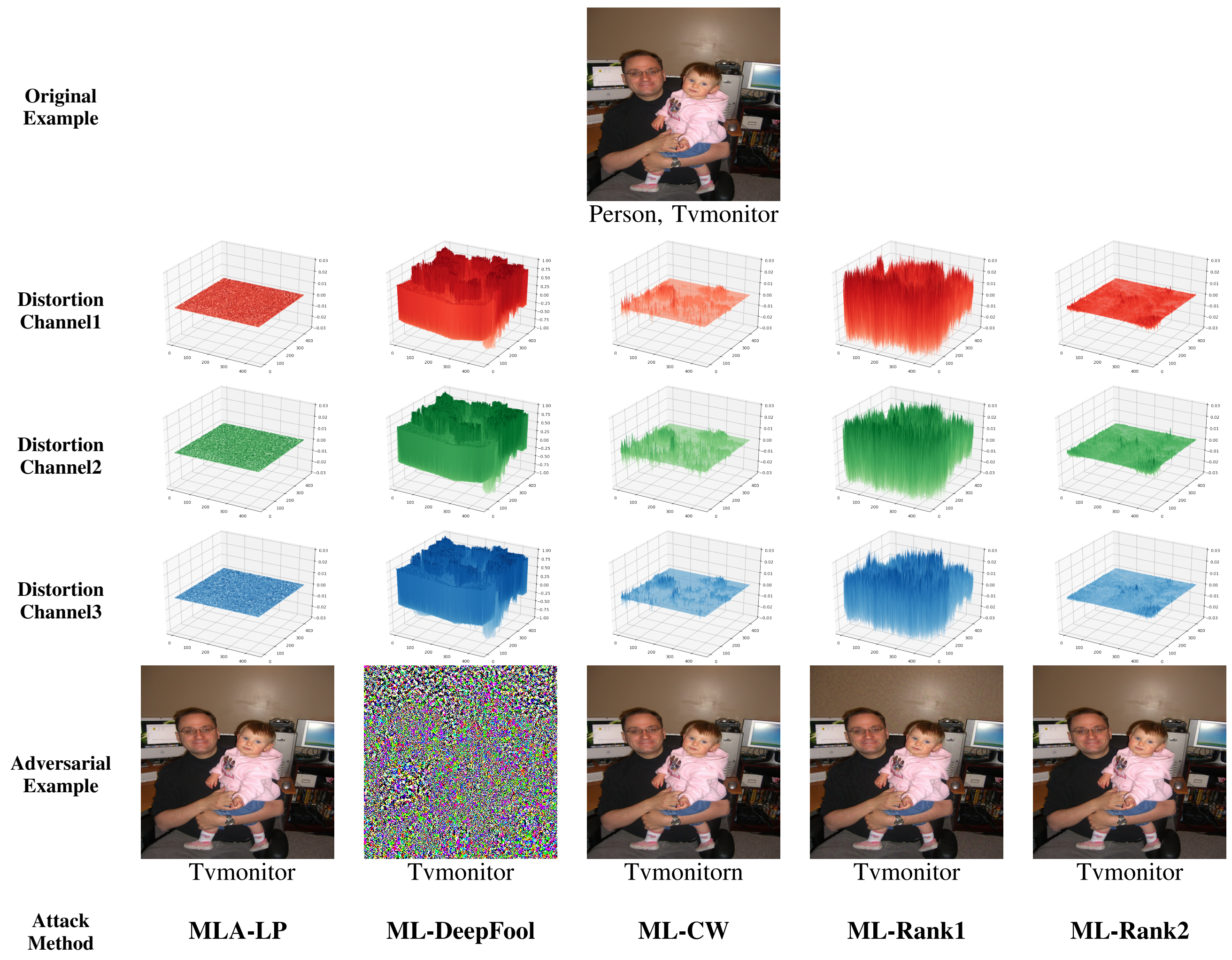

Fig. 1. Comparisons of adversarial examples and distortions generated by MLA-LP, ML-DeepFool, ML-CW, ML-Rank1, and ML-Rank2. The z-axis interval of the 3D map of the distortion generated by the ML-DeepFool algorithm is [-1, 1], while those of the others are [-0.03, 0.03].

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

[7] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[8] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, p. 333, 2011.

[9] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 279–286.

[10] J. C. Zaragoza, E. Sucar, E. Morales, C. Bielza, and P. Larranaga, "Bayesian chain classifiers for multidimensional classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[11] A. Kumar, S. Vembu, A. K. Menon, and C. Elkan, "Beam search algorithms for multilabel learning," *Machine Learning*, vol. 92, no. 1, pp. 65–89, 2013.

[12] S.-F. Chen, Y.-C. Chen, C.-K. Yeh, and Y.-C. F. Wang, "Order-free rnn with visual attention for multi-label classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] C.-Y. Hsieh, Y.-A. Lin, and H.-T. Lin, "A deep model with local surrogate loss for general cost-sensitive multi-label learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] G. Kurata, B. Xiang, and B. Zhou, "Improved neural network-based multi-label classification with better initialization leveraging label cooccurrence," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 521–526.

[15] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[16] R. Grodzicki, J. Mańdziuk, and L. Wang, "Improved multilabel classification with neural networks," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2008, pp. 409–416.

[17] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[18] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "CNN-RNN: A unified framework for multi-label image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2285–2294.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5177–5186.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[22] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[23] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 25–32.

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

[25] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[26] W. Luo, C. Wu, N. Zhou, and L. Ni, "Random directional attack for fooling deep neural networks," *arXiv preprint arXiv:1908.02658*, 2019.

[27] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 1–7.

[28] R. Taori, A. Kamsetty, B. Chu, and N. Vemuri, "Targeted adversarial examples for black box audio systems," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 15–20.

[29] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[30] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[31] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[32] M. ApS, *The MOSEK Optimizer API for Python manual. Version 9.1.*, 2019. [Online]. Available: https://docs.mosek.com/9.1/pythonapi/index.html

[33] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[34] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[35] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2013.

[36] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[37] 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*, 2016, pp. 770–778.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.