Robustness of Multi-Label Neural Networks

Julian Mour

Advisor: Dana Drachsler Cohen
Technion - Israel Institute of Technology

April 18, 2023

1 Introduction

In the last years, many works have demonstrated that deep neural networks (DNNs) are susceptible to a variety of adversarial example attacks, e.g., [? ? ? ? ? ?]. Most adversarial attacks add small changes to the input that fool the network. However, some attacks, known as L_0 attacks, focus on perturbing a very small number of pixels (e.g., one) to an arbitrary value [? ? ?]. This type of attack is very effective for concealing adversarial modification in practice, and it can provide geometrical theoretical insight into the input space of a classifier. These attacks are usually local: for each input, they perform a computation to determine the perturbation, which may take a few minutes. Furthermore, these attacks are not robust to input transformation, so each input must be processed differently.

In this thesis, we will design universal L_0 attacks, which rely on programs. The key idea is that program can be specific to input but can be executed much faster and thus will be easier compared to prior attacks. The program will determine which pixels to perturb and to which values. Unlike previous adversarial attack methods, our programs do not employ independent optimization for each image but rather find structures/rules that can be applied to every new input to identify which pixels will be perturbed and how. Our focus is on programs that have two stages. As the first stage, the goal is to limit the search space for a successful attack to a relatively small area in the input image. During the second stage, the focus is primarily on identifying the perturbation (its location and magnitude) within that small area.

Our key idea is to formulate this problem as an optimization problem in which we try to find the optimal program. Two methods have been considered in terms of the initial stage of the program. First is an evaluation search that is trained on a limited set of inputs and given a set of possible program instructions. With each successive iteration, the search seeks to improve the program based on previously generated programs. In the second technique, a deep neural network is used to predict the locations and magnitudes of the perturbed pixels. For the second stage of the program, we developed a small program that iteratively examined all possibilities in a specific area.

In our preliminary research, we implemented our optimization approach on CIFAR-10 dataset on ResNet-18 [?] and evaluated it. We compared our results to the well-known "one-pixel attack method" [?] which is our baseline. Our initial results indicate that our program generated attacks that had almost the same success rate as the baseline, but with much shorter execution times and significantly fewer queries.

In the future, we intend to improve our algorithms to ensure a greater success rate of the attacks and shorter execution times with fewer queries. Additionally, we would like to leverage our programs to provide a better understanding of the network's decision boundaries and the factors that contribute to an attack's success.

2 Problem Definition

In this section, we define the problem we address. Informally, we are interested in synthesizing a program that, given a new image, unknown during the synthesizing process, can perturb a small number of pixels in order to perform a successful untargeted attack.

Formally, Given a classifier N, a set X of n test images and correspond true classes $c_1, c_2, ..., c_n$, our goal is to synthesis a universal perturbation program P such that N classified as many perturbed images ($P(x_i)$ s.t. $x_i \in X$) as possible to some label $t_i \neq c_i$. In order to ensure that the perturbation is relatively small when applied to natural image x_i we add the constraint $||P(x_i) - x_i||_0 \leq d$. In our case, we use $d = 1, 2, .., 5$.

$$\begin{array}{ll} \max\limits_{P} & \sum_{i=1}^{n} 1_{class(N(P(x_i))) \neq c_i} \\ \text{s.t.} & \forall x_i \in X, \quad ||P(x_i) - x_i||_0 \leq d \end{array}$$

3 Our Approach

In this thesis, we will synthesize universal programs, that will generate successful universal adversarial L_0 attacks. For the first stage of the program, we consider two different approaches:

- Evolution Search - The desired program will be based on the RGB values of pixels and small spatial regions within the image. The desired program will be created using evolution search optimization [**?** **?**]. In this program, a few learnable "filters" will be used to determine the location and the magnitude of the perturbation. Each filter consists of several learnable parameters including the spatial size (1×1 or 3×3 or 5×5) of the filter, the channel (red, green or blue), its value boundaries, and the perturbation. Accordingly, our current settings determine whether a pixel is "agreed" on a specific filter if the mean value of the pixels in the selected channel in some spatial size around the pixel lies within the specified boundaries (higher/lower than some numeric threshold). A perturbation will be applied to pixels that "agree" with the greatest number of filters. Basically, the synthesized program P is a set of filters F that takes a "natural" input image x_i and calculates the magnitudes and positions of the perturbations outputs them.

- Deep Learning - By using this approach, we will build a CNN that will take an image as an input, determine the position (regression task) and the magnitude of the perturbation (classification task - each class represents a possible perturbation) as an output. The network will be trained using individual attacks on the trainset.

There are several differences between the two approaches mentioned above. Synthesized by evolution search programs may be more straightforward and interpretable than deep learning techniques. Moreover, evolution search does not require the creation of a new dataset of successful adversarial attacks. On the other hand, the deep learning approach is based on a CNN which is regarded as a strong model for identifying features in images.

During the second stage of the program, after calculating the suggested perturbation (location and magnitude) in the first stage, the program will attempt to locate an attack within the same spatial area as the suggested perturbation. The current implementation involves systematically searching around the suggested area in successive iterations, each time trying to go further from the suggested pixel.

4 Preliminary Results

We implemented our approach and evaluated it on the CIFAR-10 test dataset (consisting of images from ten classes) and ResNet-18 as the "attacked" network. We will calculate the success

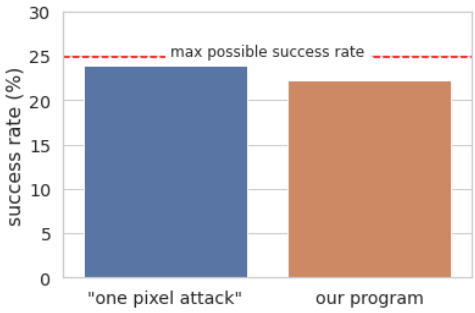

Figure 1. (a): Success Rate

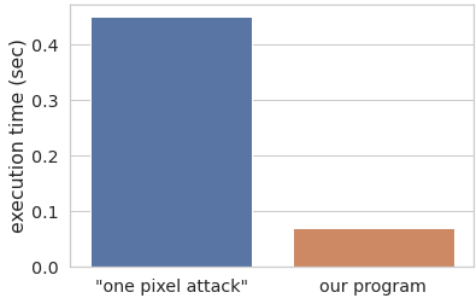

Figure 1. (b): Average Execution Times

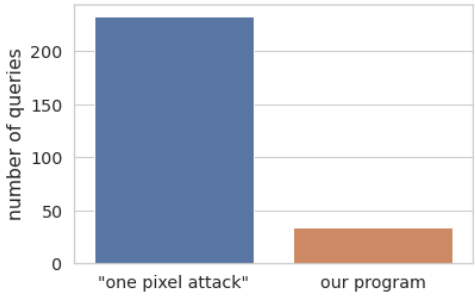

Figure 1. (c): Average Number of Queries

Figure 1: Comparison between the results of our method to the results of the "one pixel attack" method.

rate of the attacks which generated by our program to determine whether it is successful. Success rate is the percentage of the successful attacks among originally correctly classified images:

$$\text{Success Rate} = \frac{\sum_{i=1}^{n} 1_{class(N(P(x_i)))\neq c_i \wedge class(N(x_i))=c_i}}{\sum_{i=1}^{n} 1_{class(N(x_i))=c_i}} \cdot 100$$

Where $n = 10,000$ is number of test images, N is the classification CNN, P is the synthesized program, and x_i is an image from class c_i.

Using the current implementation, we achieved a success rate of 22.28% on ResNet-18 when on average each successful attack (on one image) takes 0.07 seconds and 34.32 queries. The program was restricted to only perturb one pixel per image, and to only one possible perturbation (maximum values of the three channels). We evaluated that the maximum possible success rate with our current setting (one possible perturbation of one pixel) is 24.81% when we use brute force (iterating over all possible pixels) which gives the optimal solution but is very inefficient in terms of execution time and the number of queries. Furthermore we evaluated the well-known "one-pixel attack" method [**?**] on the current setting. We have achieved a success rate of 23.96% using that attack, with an average execution time of 0.45 seconds and an average of 232.59 queries - both calculated per one input image and only for successful attacks. Figure 1 shows the comparison between the results of our method to the results of the "one pixel attack" method.

5 Future Research Objectives

In light of the preliminary work, we aim to further explore our ideas in the following directions:

- Improved algorithm: success rate, execution times and number of queries - As a result of the current implementations and settings of our two approaches, the success rate of the

attacks generated by the program is lower than our baseline. Therefore, we will implement different evolution methods, take a different approach to program instruction design, and develop a different architecture (e.g. transformers [?]) for the deep network approach in order to increase success rates. While improving the success rate, we will strive to keep execution times as short as possible with a low number of queries.

- Explainability - One possible advantage of using a universal program to generate adversarial examples is to be able to explain the network's decision boundaries and the reasons that lead to the success of attacks. As an example, by synthesizing a program, we can determine which spatial regions or channels are relevant to the success of the attack. Using the deep learning approach, we are able to use other known tools for an explanation, such as LIME [?].

6 Related Work

Our thesis lies at the intersection of adversarial attacks and program synthesis. We next review some related work.

6.1 Adversarial Attacks

Several aspects can be used to classify adversarial attacks [?]:

- Targeted Vs. Untargeted - A targeted attack [?] attempts to mislead a classifier by classifying input to a *specific class* that differs from the true class of input. In untargeted attacks [?], the goal is to cause the classifier to classify the input to *any class* that is different from the true class.

- Individual Vs. Universal - Individual attacks generate different attacks for each different input, by optimizing the attack based on the input [? ? ?]. Universal attacks, on the other hand, generate one attack for all possible inputs. Studies from the last few years have indicated that universal adversarial attacks are capable of generalizing well to unseen data. The first method for universal adversarial attack was based on accumulating perturbations iteratively by composing perturbations for each data point [?]. Other studies have used singular vectors [?] and networks for adversary generation (NAGs) [?].

- White box Vs. Black box - In white box attacks, the architecture, weights, and parameters of the model are known to the adversary. First, white box adversarial attacks were derived through the generation of artificial perturbations using gradient-based optimization [?]. There have been numerous works since then that have demonstrated different approaches to generating white-box adversarial attacks, such as FGSM[?], JSMA [?], and Deepfool [?]. Attackers using black box methods assume they can only query the model without any further knowledge of it. Several studies have shown that a black-box approach can lead to generating successful attacks [? ? ? ?].

Additionally, we investigate L_0 attacks, which aim to modify a small number of pixels. In previous studies, it has been demonstrated that even a single pixel perturbation can lead to successful attacks[? ? ?].

6.2 Program synthesis

Program synthesis is the task of automatically generating a computer program from user requirements. It has been studied for decades and shown success in a variety of fields, such as

data wrangling[?], code repair[?], code suggestions[?], probabilistic modeling[?] and superoptimization[?]. Later, constructive techniques have been developed, such as enumerative search[?], programming by examples[?], constraint solving[?], stochastic search[?], neural program synthesis[?] and genetic programming[?]. In our preliminary work, we focused on a variant of genetic search.