Verifying the Resilience of Neural Network Watermarking

Ben Goldberger[1], Yossi Adi[2], Joseph Keshet[2], and Guy Katz[1]

[1] The Hebrew University of Jerusalem, Israel
{jjgold, guykatz}@cs.huji.ac.il
[2] Bar Ilan University, Israel
{a, b}@biu.ac.il

1 Introduction

DNNs are really important and conquering the world

Learning as a service paradigm: sell your almost-trained network. What if someone tries to rip you off?

Watermarking as the solution to earlier problem. But how can we be sure watermarks are good?

DNN verification is a new and promising field. We propose a novel methodology to use verification to measure and verify the robustness of watermarking techniques. Main uses of our approach: verify watermarked networks, assess efficiency of watermarking schemes.

The rest of this paper is organized as follows. In Section 2 we provide the necessary background on DNNs, watermarking, and DNN verification. Next, in Section 3 we introduce our technique for casting the watermark resilience problem into a verification problem. Section 4 describes our implementation and evaluation of the approach on several watermarked DNNs for image recognition. We discuss related work in Section 5, and conclude in Section 6.

2 Background

[2,3]

3 Finding the minimal change that will get rid of the network's WaterMark

Given a watermarked trained neural network as described here [1]. We tested what is the minimal change to the network last layer in order to "remove" some watermarks from the network.

Guy: make this a proper citation

3.1 Defining the problem

Given a neural network N with an output size m the network decision for an input x is defined as the coordinate with the maximal value, if the network output is the vector y the decision is $argmax_{i\in[m]}\left\{y_i\right\}$

Given a watermarked network N with a set of K watermarks (A set of inputs to the network) $\{x_1,\cdots,x_K\}$ we'll mark the network last layer L such that L is a $m\times n$ matrix were n is the layer's number of neurons and m is the network output size. The change to the last layer will be a matrix with the same dimension as L we'll mark as ε, such that $\varepsilon_{i,j}$ is the change to the last layer matrix entry $L_{i,j}$. Well measure the overall change to the layer as $\|\varepsilon\|_\infty = max_{i,j}\left\{|\varepsilon_{i,j}|\right\}$.

For a certain input x we're only interested in the input to the last layer we'll mark the input to the last layer v. v is a $n\times 1$ vector. So the original network output $y=Lv$ and the changed network output is $y'=(L+\varepsilon)v$. For a single input x we need to find the minimal ε so that the $argmax_{i\in[m]}\left\{y_i\right\}\neq argmax_{i\in[m]}\left\{y'_i\right\}$

Denote $d := argmax_{i\in[m]}\left\{y_i\right\}$
For some $d'\in[m]\,, d'\neq d$ finding ε with minimal $\|\varepsilon\|_\infty$ such that $y'=(L+\varepsilon)v$ and $d' = argmax_{i\in[m]}\left\{y'_i\right\}$ can be described in a Linear Programming form like so:

$$
\begin{aligned}
Minimize:\quad & c\\
Subject\ to:\quad & \forall i,j \quad -c\leq\varepsilon_{i,j}\leq c\\
& y'=(L+\varepsilon)v\\
& y'_d\leq y'_{d'}
\end{aligned}
$$

*The variables are the entries in ε and y'

Using the same method we can find how to change the network to more than one input.
Given inputs $x_1,\cdots,x_k$ and their respective inputs to the last layer $v_1,\cdots,v_k$ and their respected outputs and decisions $\{y_1,\cdots,y_k\}$ $\{d_1,\cdots,d_k\}$ we want to find ε such that
$\forall 1\leq j\leq k \quad d_j\neq argmax_{i\in[m]}\left\{\left(\left(L+\varepsilon\right)v_j\right)_i\right\}$
Assuming we choose our new desired output $\{d'_1,\cdots,d'_k\}$ And now our LP looks like this:

$$
\begin{aligned}
&\mathit{Minimize:} && c\\
&\mathit{Subject\ to:} && \forall i,j \quad -c \le \varepsilon_{i,j} \le c\\
& && \forall j \quad y'_j = (L+\varepsilon)v_j\\
& && \forall j \quad \left(y'_j\right)_{d_j} \le \left(y'_j\right)_{d'_j}
\end{aligned}
$$

*The variables are the entries in ε and y'_j

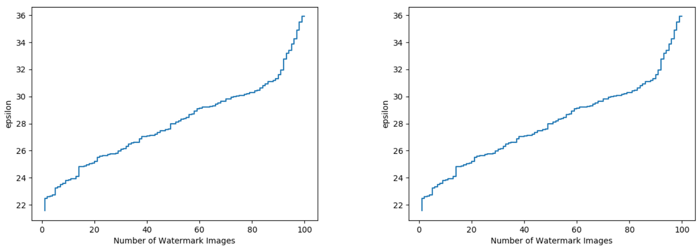
Using Marabou we can solve for the minimal ε under different norm $\|\varepsilon\|_1 = \sum_{i,j} |\varepsilon_{i,j}|$ since using this norm gives us a piece-wise linear problem.

Guy: Overall, this looks good. Some stuff will need to be moved to other sections according to the paper layout.

4 Evaluation

We tested this approach on a Neural network trained on the MNIST data set, the network was watermarked with 100 images of Gaussian noise.

Guy: I think you can start populating this section next. We will need graphs and pictures.



(a) Coffee.　　(b) More coffee.

Fig. 1: The same cup of coffee. Two times.

5 Related Work

6 Conclusion and Future Work

References

1. Y. Adi, C. Baum, B. Pinkas, and J. Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *Proc. 27st USENIX Security Symposium*, 2018.

2. G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proc. 29th Int. Conf. on Computer Aided Verification (CAV)*, pages 97–117, 2017.
3. G. Katz, D. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. Dill, M. Kochenderfer, and C. Barrett. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Proc. 31st Int. Conf. on Computer Aided Verification (CAV)*, 2019.