Maximal Robust Layer-Neighborhoods in Multi-Label Neural Networks

Julian Mour

Advisor: Dana Drachsler Cohen
Technion - Israel Institute of Technology

May 1, 2023

1 Introduction

Multi-label image classifiers are successful in various tasks such as Image Tagging [**?**], Object Detection [**?**], Facial Expression Recognition [**?**] etc. A multi-label image classifier is a model that assigns multiple labels or tags to an image to describe its contents or attributes. However, many works have demonstrated that deep neural networks (DNNs) are susceptible to adversarial example attacks, e.g., [6, 10, 15, 16, 19, 11]. These attacks add a small perturbation to a correctly classified input with the goal of causing the network to misclassify. In particular, several works have shown the vulnerability of multi-label image classifiers [**?** **?** **?**]. To understand the robustness level of image classifiers, many verifiers have been introduced **Dana: add many citations**. However, no verifier analyzes the robustness level of multi-label classifiers.

Challenges Part of the challenge is defining what robustness means in multi-label classifiers. For (single label) classifiers, a popular definition is *local robustness*. At high-level, given an image classifier, an input to the classifier, and a perturbation limit, the classifier is locally robust if perturbing the given input up to the given limit does not change the network's classification. **Dana: explain why the straight forward extension of the definition is problematic**. Even given a suitable definition, verifying multi-label classifiers is challenging because they tend to be deeper than single label classifiers**Dana: true?**. **Dana: are there other challenges?**

Multi-label classifier robustness In this thesis, we propose a new definition for local robustness of multi-label classifiers. At high level, given a multi-label classifier, an input containing several objects and a perturbation limit, the network is robust if **Dana: complete as in the previous sentence**. T his definition allows one to understand: **Dana: complete the motivation to this definition**.

Problem definition Given **Dana: complete** our goal is to compute**Dana: complete**. A naive optimal algorithm **Dana: complete**. However, it is highly time consuming.

Key idea To scale the analysis, our key idea is to rely on oracle-guided synthesis**Dana: add citation**. Namely, we propose an algorithm that iteratively expands the specification, given by the series of epsilons**Dana: make sure it's defined in the problem definition paragraph**. At each iteration, a specification is submitted to an existing verifier, capable of analyzing**Dana: complete**. Then, based on its response, we update the specification by numerical optimization, acting as the specification synthesizer. The synthesizer's challenges are: (1) the specifications

do not adhere to partial order and (2) computing the gradient is challenging because**Dana: complete**. To cope, we propose to **Dana: complete**.

Preliminary results In our preliminary research, we implemented a basic version of the above approach. We evaluate it on the DOUBLE-MNIST test dataset**Dana: add citation**. We ran our program on three different CNN multi-label DOUBLE-MNIST classifiers that were trained differently: without a defense, with an L_0-based defense**Dana: add citation** and with the PGD defense**Dana: add citation**. Results show that **Dana: complete**.

Future goals As part of the thesis, we intend to improve our algorithm by reducing the number of queries to the verifier, and thereby shortening the execution time. To this end, we plan to **Dana: complete how you'll do it**. We also plan to compute wider robust layer-neighborhoods by **Dana: compelte how you think you'll do it**.

Dana: below is the old text need to be rewritten as part of previous paragraphs

We first build layers around the non-target object in the image. Each layer can be perturbed by an epsilon (defined specifically for this layer), while maintaining robustness for the target class - not affecting the classification of the target class. Meaning, we want a sequence of epsilons that for a specific classifier and image, defines a robust layer-neighborhood for the target class. To get best results, we want the layer-neighborhood to be maximal. **Dana: this mixes definition and algorithm – need to separate**

Our key idea is to compute the sequence of epsilons by verifying and optimizing it in each step, rather than computing each epsilon individually. Given a sequence of epsilons representing an epsilon of perturbation per layer, we check if the layer-neighborhood of a specific image is robust for the target object by running a robustness verifier on it and a specific classifier. To optimize a robust layer-neighborhood, we compute some gradients in which we can expand our neighborhood and try to keep it robust. Similarly, we might need to shrink a non-robust layer-neighborhood so that it becomes robust. The shrinking process can be done in several ways, such as defining a shrinking weight for each layer. Weights can be fixed (defined by the index of the layer) or not (e.g. sensitivity weights).

2 Problem Definition

In this section, we define the problem we address.

Informally, we are interested in developing an efficient algorithm that given a single image, it returns a robust layer-neighborhood for a target object (target class); A robust sequence of epsilons each representing the perturbation per layer. Each layer is defined to be a set of pixels that share the same distance from a specific non-target object (non-target class) in the image.

Formally, Given a classifier F, an image x of size $n \times m$ containing two objects: c_{target} and $c_{non-target}$ s.t. $F(x) = \{c_{target}, c_{non-target}\}$, we define the set of layers to be:

$$\mathrm{L}_x = \{l_0^x, l_1^x, \ldots, l_r^x\}$$
$$\text{s.t. } \mathrm{l}_d^x = \{(i,j) \in [1,\mathrm{n}] \times [1,m] \mid dist((i,j), c_{non-target}) = d\}$$

We aim to build a program that returns a sequence of epsilons $(\varepsilon_0^x, \varepsilon_1^x, \ldots, \varepsilon_r^x)$ representing a robust perturbations of the pixels in the corresponding layers for the target class. Formally:

$$\varepsilon_x = (\varepsilon_0^x, \varepsilon_1^x, \ldots, \varepsilon_r^x)$$
$$\text{s.t. } \forall x' \in N(x) : c_{target} \in F(x')$$
$$N(x) = \{x' \mid \forall l_d^x \in L_x : \forall (i,j) \in l_d^x : x'_{i,j} \in [x_{i,j} - \varepsilon_d^x, x_{i,j} + \varepsilon_d^x]\}$$

3 Our Approach

In this thesis, we will build a program that given a multi-label classifier and an image, can find a layer-neighborhood of the image around a certain object (non-target class) that is robust for another object (target class) in the image. A straight-forward solution would be to calculate each epsilon individually using a verifier and binary search. Problem is this approach is very expensive in time hence inefficient. Our approach aims to reach similar results as the straight-forward one, but in much less time. In principle, our approach is similar to MaRVeL's [8]. The desired program will have two main components:

- The Verifier - The verifier's main job is to check whether a given layer-neighborhood $N(x)$ of an image is robust for the classifier and pass the result to the optimizer. Since this is a multi-label classifier, we are interested only in the target class robustness; The verifier will say that the neighborhood is robust if and only if all images in the neighborhood are classified to the target class. The verifier in our case is a MILP based on the MIPVerify program [18] that translates the robustness problem of a multi-label Neural-Network to a MILP maximization problem and uses the Gurobi Optimizer to solve it. In addition to checking whether the neighborhood is robust or not, the verifier returns the weakest points of the image. These are the points in the checked neighborhood that are least robust and the verifier will pass them to the optimizer as well.

- The Optimizer - The optimizer's job is to update a given robust layer-neighborhood by expanding it more, as we intend to maximize the norm of the neighborhood. In practice, we try to maximize the norm of the neighborhood and the Robustness Level (RL) of the classifier as well. This is done by computing the gradients of both components; The norm's gradient is computed in a straightforward way while the RL's gradient is computed using the weakest points found by the verifier. At last, we calculate the step gradient, which is a combination of both previous gradients, and expand the neighborhood towards the step gradient. If the neighborhood checked by the verifier isn't robust, then we skip the optimization part. Instead, we try to minimally shrink the neighborhood so that its robust. We do that in two different approaches:

  - Fixed weights - We shrink the neighborhood using fixed weights, where we aim to shrink the neighborhood more in layers that are far from the non-target object and less in layers that are close to it.
  - Sensitivity weights - Similar to the fixed weights approach, but we aim to shrink more in layers that are less sensitive to perturbations. We get the sensitivity weights of an image by using the Vanilla Gradient method, introduced by Simonyan et al. [14].

  The new layer-neighborhood will be the input of the verifier in the next iteration.

Like that, we do this in iterations many times until convergence of the layer-neighborhood and the RL. We aim to reach a maximal robust layer-neighborhood at the end of the process.

4 Preliminary Results

We implemented our approach and ran it on part of the DOUBLE-MNIST test dataset, consisting of images from ten classes; $C = \{0, 1, \ldots, 9\}$ (the 10 different digits), each image classified to two different classes (contains two different digits). An example of a test dataset sample is shown in Figure 1, where the digit 4 is the target object and the digit 9 is the non-target object.

We ran our program on 3 different CNN multi-label DOUBLE-MNIST classifiers. While the three of them solve the same classification problem, they differ in their training process:

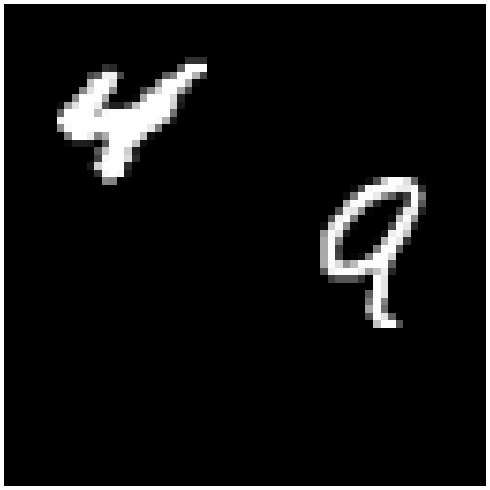

Figure 1: DOUBLE-MNIST sample

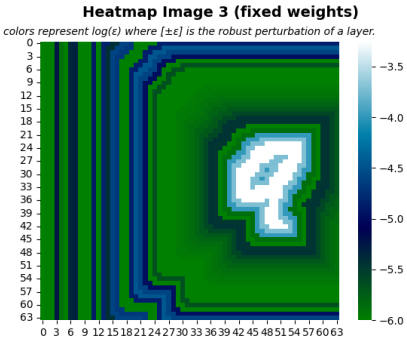

Figure 2. (a): fixed weights

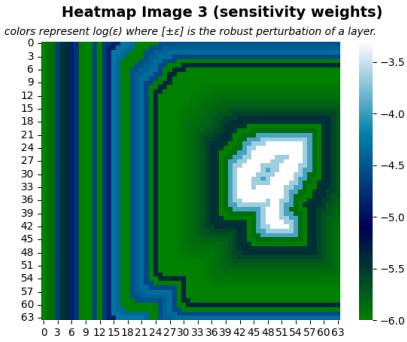

Figure 2. (b): sensitivity weights

Figure 2: No Defense

- Network with no defense - This network is trained regularly on the original DOUBLE-MNIST training dataset, without additional processing done to the training dataset. In Figure 2 we present results of our program ran on this classifier and a single image, as a heatmap representing the epsilons per each layer.

- Network with L_0 defense - This network is trained with L_0 defense. To achieve this kind of defense, we add data-augmentation to the process of training; Before forwarding a training sample to the network, we add random noise to the image. Specifically, we randomly choose a rectangular section in the image and zero all its pixels (color them black). In Figure 3 we present results of our program ran on this classifier and a single image, as a heatmap representing the epsilons per each layer.

- Network with L_∞ defense - This network is trained with L_∞ defense. To achieve this kind of defense, we also use data-augmentation, but this time we use PGD (Projected Gradient Descent) [2]. This also involves training the model with adversarial examples, but unlike the L_0 defense, the added perturbations can be anywhere in the input. The PGD generated adversarial examples are achieved by trying to increase the model's loss function as much as possible. Therefore, training the model with such adversarial examples should make the network more robust.

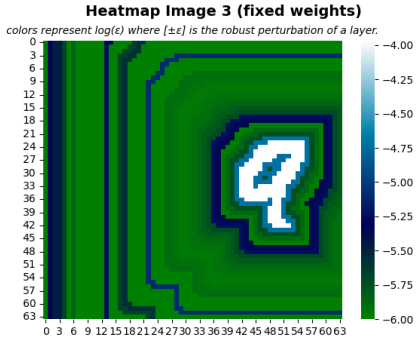

Figure 3. (a): fixed weights

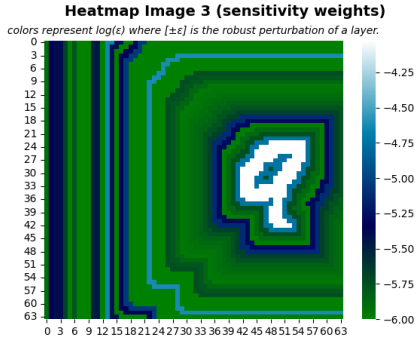

Figure 3. (b): sensitivity weights

Figure 3: L_0 Defense

5 Future Research Objectives

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

- Improved algorithm: Maximizing the neighborhood norm, execution time and number of queries - The main challenge in using our above-mentioned approach is the calculation of the weakest points that are later used by the optimizer to compute the RL gradient. Where in the regular single-label case at most $|C|$ queries are executed per iteration to achieve an accurate and unbiased RL gradient, this number jumps to $|C|^2$ in the multi-label case if the same approach as MaRVeL's is used, leading to a much longer execution time. Also, we aim to maximize the neighborhood norm; We want to achieve wide robust layer-neighborhoods. This can be affected by many factors (e.g. the shrinking method used in the algorithm) - we will explore each of these factors and look for the best methods to achieve this goal.

- Explainability - The goal of our program is to present a relation between two objects in an image for a specific multi-label classifier. The results can tell us how much and where we can change in one object so that it doesn't or does affect the other. These can vary between different classifiers as well, which will also help us understand which type of multi-label networks are most vulnerable to perturbations in different locations in their inputs.

6 Related Work

Our thesis topics mainly involve Neural-Networks Verifiers, Adversarial Attacks in Multi-Label Classification and Counter Example Guided Synthesis. We next review some related work.

6.1 Neural-Networks Verifiers

Neural network verifiers are tools or methods that are used to ensure the robustness of neural networks. These verifiers use mathematical techniques such as constraint solving and model checking to analyze the behavior of neural networks and ensure that they operate correctly under certain kinds of adversarial attacks. There are various neural network verification techniques, such as abstract interpretation (e.g. [4, 17]), linear programming (e.g. [18]) and more. Verifiers are usually divided into two categories:

- Incomplete Verifiers - an incomplete verifier may only be able to prove the absence of adversarial examples for a subset of the inputs (not all of them) [17, 5].

- Complete Verifiers - a complete verifier can prove the absence of adversarial examples for all inputs [18, 9], but is likely more time expensive than other incomplete verifiers.

In our preliminary work we used $MIPVerify$, a MILP (Mixed Integer Linear Programming) verifier [18] which is a complete verifier.

6.2 Adversarial Attacks in Multi-Label Classification

Adversarial attacks in multi-label classification refer to the phenomenon where an adversary intentionally modifies the input data to a multi-label classifier in order to cause misclassification or mislabeling of the input. In multi-label classification, each input can be assigned multiple labels, and the classifier is trained to predict a subset of these labels for each input. Adversarial attacks in this context can take various forms, such as adding perturbations to the input data to cause the classifier to predict any different incorrect subset of labels, or to cause the classifier to not predict a specific correct label. We focus on the latter. Most existing works on adversarial examples have been focused on the case of multi-class single-label classification [1, 3, 7, 12], but few on the case of multi-label classification [13].

6.3 Counter Example Guided Synthesis

Counter example guided synthesis (CEGIS) is a technique used in formal verification and program synthesis to generate correct programs or system designs from specific given specifications. The basic idea behind CEGIS is to iteratively search for a candidate that satisfies the specification, while using counterexamples to refine the search space and guide the synthesis process. In this thesis we use CEGIS to find the desired epsilons sequence that will define a robust maximal layer-neighborhood; Our specification - a maximal robust layer-neighborhood, the counterexamples - adversarial examples and weakest points. We submit several queries to a verifier in each iteration and update the epsilons sequence accordingly, given the counterexamples. Previous work also used CEGIS to find maximal robust neighborhoods [8], we use similar approach as this work.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE*, 2018.

[2] Ludwig Schmidt-Dimitris Tsipras Adrian Vladu Aleksander Madry, Aleksandar Makelov. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018.

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *IEEE*, 2017.

[4] Markus Püschel-Martin Vechev Gagandeep Singh, Timon Gehr. An abstract domain for certifying neural networks. *ACM*, 2019.

[5] Matthew Mirman-Markus Püschel Martin Vechev Gagandeep Singh, Timon Gehr. Fast and effective robustness certification. *NeurIPS*, 2018.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In ICLR*, 2015.

[7] Jonathon Shlens Ian J Goodfellow and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

[8] Anan Kabaha and Dana Drachsler Cohen. Maximal robust neural network specifications via oracle-guided numerical optimization. *VMCAI*, 2023.

[9] Shiqi Wang-Yihan Wang Suman Jana Xue Lin Cho-Jui Hsieh Kaidi Xu, Huan Zhang. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers, 11 2020.

[10] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *In ICLR*, 2017.

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *In ICLR*, 2018.

[12] Alhussein Fawzi Seyed-Mohsen Moosavi-Dezfooli and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[13] Xin Wang-Siwei Lyu Shu Hu, Lipeng Ke. Tkml-ap: Adversarial attacks to top-k multi-label learning. *ICCV*, 2021.

[14] Andrea Vedaldi Simonyan, Karen and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps., 2013.

[15] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[16] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. *In IJCNN*, 2016.

[17] Timon Gehr; Matthew Mirman; Dana Drachsler-Cohen; Petar Tsankov; Swarat Chaudhuri; Martin Vechev. Safety and robustness certification of neural networks with abstract interpretation. *IEEE*, 2018.

[18] Russ Tedrake Vincent Tjeng, Kai Xiao. Evaluating robustness of neural networks with mixed integer programming. *ICLR*, 2019.

[19] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *In IEEE Trans. Neural Networks Learn. Syst*, 2019.