Robustness of Multi-Label Neural Networks

Julian Mour

Advisor: Dana Drachsler Cohen
Technion - Israel Institute of Technology

May 9, 2023

1 Introduction

A multi-label image classifier assigns to an input image a set of labels to describe its contents. These classifiers are successful in various tasks, such as image tagging [19], object detection [20], and facial expression recognition [23]. However, many works have demonstrated that deep neural networks (DNNs) are susceptible to adversarial example attacks, e.g., [6, 15, 27, 28, 32, 18]. In particular, several works have shown the vulnerability of multi-label image classifiers [21, 11, 25]. These attacks add a small perturbation to a correctly classified input with the goal of causing the network to misclassify. To understand the robustness level of (single-label) image classifiers, many verifiers have been introduced [30, 29, 5, 14, 4]. 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. The set of perturbed inputs is called the input's *neighborhood*. In multi-label classification, where inputs are assigned to several labels, local robustness can be defined in various ways. For example, one can require that a network is robust if all labels remain or alternatively require only a subset of labels to remain unchanged. Even given a suitable definition, verifying multi-label classifiers is challenging because they tend to be deeper and more complex than single-label classifiers.

Multi-label robustness In this thesis, we focus on multi-label classifiers that take as input images showing multiple objects. For example, an image showing a road with traffic signs, cars and pedestrians, where the classifier's goal is to detect the objects in the image. For this setting, we propose a new attack model and a corresponding local robustness property. The attack model assumes an attacker can manipulate some objects and their surrounding, with the goal of causing the classifier to miss some target objects (e.g., pedestrians). The corresponding property aims to quantify how large the perturbation of the manipulated objects can be without affecting the target object's classification and how large their manipulated surrounding can be. Namely, the property aims to maximize the perturbation size and the manipulated objects' surrounding area. This definition allows one to understand how changing a specific object in a multi-labeled image affects another's classification in a multi-label classifiers and identify their robustness relation. For simplicity's sake, we assume the image shows two objects, the target object and the perturbed object. Formally, we model the neighborhood by a sequence of epsilons, each corresponds to the maximal allowed perturbation in its respective layer. The first epsilon corresponds to the pixels at the perturbed object, the next one to the pixels immediately surrounding the perturbed object and so on. We further assume a weight vector, assigning a weight for each layer. The goal is to compute the series of epsilons maximizing the weighted sum. This problem is highly challenging for two reasons: It is a multi-dimensional search space and verifying that an epsilon vector belongs to this space (i.e., it represents a robust neighborhood) requires to invoke a (standard) local robustness verifier, which takes a non negligible time.

Key idea To scale the analysis, our key idea is to rely on oracle-guided synthesis [9]. Namely, we propose an algorithm that iteratively expands a given sequence of epsilons, corresponding to a robust neighborhood. At each iteration, an epsilon sequence is submitted to an existing local robustness verifier. Based on its response, we update the epsilon sequence by numerical optimization. To this end, we propose to define gradients from the verifier's output.

Preliminary results In our preliminary research, we implemented a basic version of the above approach. We evaluate it on the DOUBLE-MNIST test dataset [10]. We evaluate our algorithm on three different CNN multi-label DOUBLE-MNIST classifiers that were trained differently: without a defense, with an L_0-based defense [26] and with the PGD defense [2]. Results show that the latter model is the most robust. **Julian: check after getting PGD results**.

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 use faster verifiers for some of the queries. We also plan to rely on sensitive layers, to dynamically update the weight vector with the goal of identifying larger robust neighborhoods.

2 Problem Definition

In this section, we define the problem we address. For simplicity's sake, all our definitions and algorithms focus on images with two objects – the target one and the perturbed one – but they can extend to multiple objects. Informally, given a multi-label classifier, an image showing a target object, we aim to compute a robust layer-neighborhood, given for a target object (target class). The neighborhood is defined by a sequence of epsilons representing the perturbation per layer. We begin with definitions and then define our problem. A multi-label classifier is a function mapping an instance, in our case an image from an input domain $\mathbb{R}^{n\times m}$, to a score vector over the possible set of classes C, $F:\mathbb{R}^{n\times m}\rightarrow\mathbb{R}^{|C|}$. Given an image x, the classification C' of a multi-abel classifier F for x is the subset of k classes with the highest scores, $C'=\{\arg i^{th}\max_{c'\in C}F(x)_{c'}\}_{i=1}^{k}$. In our case and for simplicity's sake $k=2$. A neighborhood of input x is a set of inputs $N(x)\subseteq\mathbb{R}^{n\times m}$ containing x. A neighborhood $N(x)$ is robust if all inputs in it are classified to a **target label** $c_t\in C$, $\forall x'\in N(x): c_t\in\{\arg i^{th}\max_{c'\in C}F(x)_{c'}\}_{i=1}^{k}$. We focus on layered neighborhoods. To define it, we denote the set of pixels showing an object o in an image x by P_o^x. Given an image x showing two objects o_t and o_{nt}, the target one with class c_t and the perturbed one with class c_{nt} and its set of pixels $P_{o_{nt}}^x$, the d^{th} layer is the set of pixels that their Chebyshev distance (L_∞) from $P_{o_{nt}}^x$ is d. Given two pixels $p=(i,j),p'=(i',j')\in[n]\times[m]$, their distance is $dist(p,p')=||p-p'||_\infty=\max\{|i-i'|,|j-j'|\}$. Given a pixel $p=(i,j)\in[n]\times[m]$ and a set of pixels $P\subseteq[n]\times[m]$, their distance is the minimum distance of p to any pixel in P: $dist(p,P)=\min\{dist(p,p')\mid p'\in P\}$. Given an image x, an object o, and a distance d, we define the d^{th} layer: $l_d^{x,o}=\{p\in[n]\times[m]\mid dist(p,P_o^x)=d\}$. Given an image x and a non-target object o_{nt}, we define the set of layers by $L_x^{o_{nt}}=\{l_0^{x,o_{nt}},l_1^{x,o_{nt}},\ldots,l_r^{x,o_{nt}}\}$, where $r+1$ is the number of layers around o_{nt} covering all pixels in x.

A layered neighborhood Given an image x, the non-target object's pixels $P_{o_{nt}}^x$ and a series of maximal allowed perturbation for every layer $\epsilon=(\epsilon_0,\ldots,\epsilon_r)$, a layered neighborhood $N_\epsilon^{o_{nt}}(x)$

is the set of all images whose perturbation at layer d is bounded by the respective perturbation limit:

$$N_{\epsilon}^{o_{nt}}(x) = \{x' \in \mathbb{R}^{n\times m} \mid \forall 0 \leq d \leq r+1\ \forall (i,j) \in l_d^{x,o_{nt}} : |x'_{i,j} - x_{i,j}| < \epsilon_d\}$$

We also denote the set of all robust layered neighborhoods of an image x with RLN_x, and the set of epsilon sequences representing them with RES_x.

Given a weight vector $w = (w_0, w_1, \dots, w_r)$ assigning a weight for each layer, the size of a layered neighborhood $N_{\epsilon}^{o_{nt}}(x)$ is $||N_{\epsilon}^{o_{nt}}(x)|| = w \times \epsilon^T = \sum_{d=0}^{r} w_d \cdot \epsilon_d$.

Now we can define our problem: Given a classifier $F : \mathbb{R}^{n\times m} \to \mathbb{R}^{|C|}$, an image $x \in [0,1]^{n\times m}$ containing two objects: o_t and o_{nt} s.t. $C' = \{\arg i^{th} \max_{c'\in C} F(x)_{c'}\}_{i=1}^{k} = \{c_t, c_{nt}\}$, the goal is to compute a sequence of epsilons $\epsilon^* = ({\epsilon_0}^*, {\epsilon_1}^*, \dots, {\epsilon_r}^*)$ satisfying:

1. F is robust at $N_{\epsilon^*}^{o_{nt}}(x)$.
2. For every ϵ' expanding ϵ^*, $N_{\epsilon'}^{o_{nt}}(x)$ is not robust.
3. $N_{\epsilon^*}^{o_{nt}}(x)$ maximizes its size among all layered neighborhoods meeting (1) and (2).

3 Our Approach

In this thesis, we will design an algorithm that given a multi-label classifier and an image computes the maximal neighborhood given by an epsilon sequence, describing how perturbation of the non-target object affects the classification of the target object. A naive algorithm computes the epsilon series one by one, where at each iteration it computes the maximal epsilon using a binary search. However, this approach is suitable in case the weight vector poses a strict ordering on the importance of the layers, and it also has a high time overhead. Instead, we aim to build on the oracle-guided numerical verification proposed in [12], to obtain a scalable algorithm. Technically, our algorithm has three main components, that iteratively interact with one another:

- A local robustness verifier: Given a classifier and a layer-neighborhood I of an image x, it determines whether the classifier is robust in this neighborhood.
- A numerical optimizer: Given a classifier and a robust layer-neighborhood, it attempts to expand the neighborhood into a larger robust neighborhood, with respect to the weight vector.
- A counterexample-guided inductive synthesiser (CEGIS): Given a classifier and a *non-robust* layer-neighborhood, it attempts to identify directions of the previously robust neighborhood that cannot be further expanded.

The components interact until the neighborhood cannot be further expanded. We next provide details about the different components and explain the open challenges.

The verifier There are many verifiers that can reason about the local robustness of a (single-label) classifier. However, none addresses a multi-label classifier. Thus, part of the challenge is adapting a verifier to multi-label classification. In particular, the verifier only has to prove that one of the labels is the target object's label. In our preliminary research, we rely on the mixed-integer linear program (MILP) based verifier, called MIPVerify [30]. This verifier encodes the robustness task into a MILP maximization problem and uses the Gurobi Optimizer to solve it. For an input x, a neighborhood $I(x)$ and classifier F, the main task is to prove or refute robustness. The MILP objective represents the difference between the highest score of an incorrect class and the score of the correct class (c_{target}) of a classifier F, and the MILP solution is an input x' that maximizes it.

$$\max_{x' \in I(x)} \{max_{c \in C, c \neq c_{target}} \{F(x')_c\} - F(x')_{c_{target}}\}$$

A negative value of the maximized objective indicates a robust neighborhood. We adapt it to multi-label classifiers by changing the objective. Instead of picking the highest score, we now consider the second-highest score. Meaning that the objective now represents the difference between the second-highest score of an incorrect class and the score of the target class (c_{target}).

$$\max_{x' \in I(x)} \{2^{nd} max_{c \in C, c \neq c_{target}} \{F(x')_c\} - F(x')_{c_{target}}\}$$

A negative value of the maximized objective indicates a robust neighborhood in the multi-label case. Beyond determining whether a neighborhood is robust or not, the verifier returns a set of points maximizing the objective, to which we call *the weakest points of the image.* These points later help the optimizer to identify robust directions to expand the current neighborhood.

The optimizer Our optimizer expands a robust neighborhood by computing the gradient of the optimization problem defined in the previous section. Since it is a constrained optimization, we relax the constraint and add an equivalent term to the optimization goal, as standard. We call this term *the robustness level* (RL). To expand a robust neighborhood, our optimizer computes the gradients of both terms. 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. Given the gradients, the optimizer expands the neighborhood by a small step and submits to the verifier.

The CEGIS component The CEGIS components takes a non-robust neighborhood and the previously robust neighborhood and attempts to identify directions of the previously robust neighborhood that cannot be further expanded. Computing the exact directions requires an exponential number of queries to the verifier. Instead, we want shrink the non-robust neighborhood in the direction of the previously robust neighborhood. We shrink each layer according to an assumed weight - the cutting weight. The layer's cutting weight determines how much should the layer be shrunk towards the previously robust neighborhood. We notate the cutting weights vector with cw_x, and its complement vector with cw_x^* and define it as follows:

$$cw_x = (cw_0^x, cw_1^x, \ldots, cw_r^x)$$
$$cw_x^* = (1 - cw_0^x, 1 - cw_1^x, \ldots, 1 - cw_r^x)$$

Mathematically, this translates to computing a new epsilon sequence ε_x' that is a weighted average of the current non-robust epsilon sequence ε_x and the previously robust neighborhood ε_x^*. We use *element-wise multiplication* ($\odot$) to multiply each element in the epsilon sequences with its corresponding weight in the weights vectors:

$${\varepsilon_x}' = \varepsilon_x \odot cw_x^T + \varepsilon_x^* \odot {cw_x^*}^T$$

We consider two 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:

$$cw_x = (\frac{r}{r+1}^5, \frac{r-1}{r+1}^5, \frac{r-2}{r+1}^5, \ldots, 0)$$

Figure 1: DOUBLE-MNIST sample

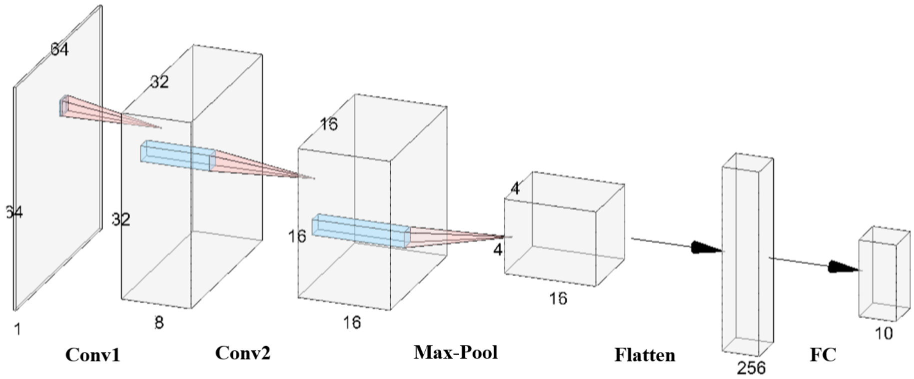

Figure 2: Classifiers' architecture - 2 Convolutional layers, 1 Max-Pool layer and 1 FC layer

- 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. [24]. This method can be used to find sensitive pixels in an image by simply computing the gradient of the loss function with respect to the image pixels using backpropagation. Pixels with large gradients are likely to be more sensitive to perturbations, and perturbing them is likely to affect the classifier's robustness more. We translate the problem of finding sensitive pixels to finding sensitive layers by averaging all pixels' gradients in each layer, and achieve the sensitivity weights vector $cw_x = (sw_0^x, sw_1^x, \ldots, sw_r^x,)$ where $sw_i^x < sw_j^x$ iff layer i is more sensitive to perturbations than layer j.

4 Preliminary Results

We evaluated our preliminary approach on the DOUBLE-MNIST test dataset, consisting of images showing two digits. The multi-classifier's goal is to return the correct two digits. An example of an image is shown in Figure 1. We ran our algorithm on three different CNN multi-label DOUBLE-MNIST classifiers, all with the same architecture (Figure 2) but a different training procedure:

- Without defense.
- With an L_0 defense: this defense relies on the following data augmentation. Before forwarding a training sample to the network, we add random noise to the image in the form of a black rectangle.

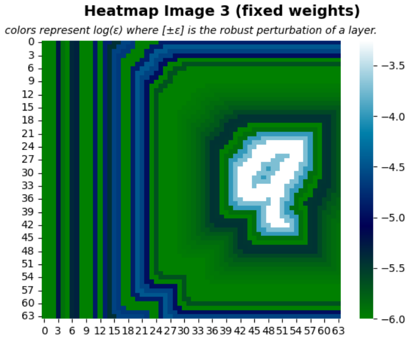

Figure 3. (a): fixed weights

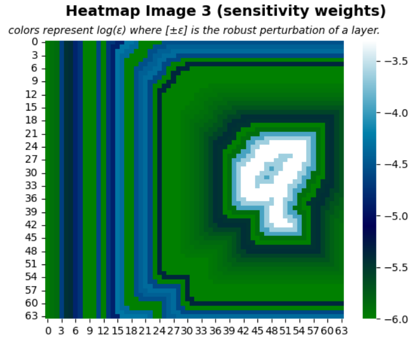

Figure 3. (b): sensitivity weights

Figure 3: No Defense

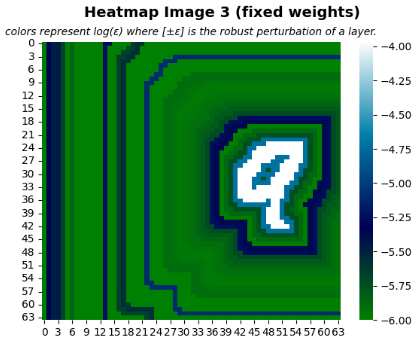

Figure 4. (a): fixed weights

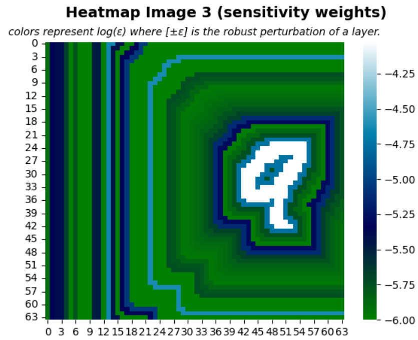

Figure 4. (b): sensitivity weights

Figure 4: L_0 Defense

- With an L_∞ defense: using the Projected Gradient Descent (PGD) defense [2]. This defense also involves training the model with adversarial examples, but unlike the L_0 defense, the added perturbations are a small value and can be anywhere in the input.

In Figure 3 we present results of our program ran on a classifier trained without defense, and a single image, as a heatmap representing the epsilons per each layer. In Figure 4 we present results of our program ran on a classifier trained with L_0 defense, and a single image, as a heatmap representing the epsilons per each layer.

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, 29]), linear programming (e.g. [30]) 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) [29, 5].
- Complete Verifiers - a complete verifier can prove the absence of adversarial examples for all inputs [30, 14], but is likely more time expensive than other incomplete verifiers.

In our preliminary work we used $MIPVerify$, a MILP (Mixed Integer Linear Programming) verifier [30] 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 attacks have been focused on the case of multi-class single-label classification [1, 3, 8, 22]. Several others on the case of multi-label classification, where attacks are mainly divided to two types: **(1) Targeted Attacks** - aim to bring specific incorrect labels' scores to be the highest and **(2) Untargeted Attacks** - aim to bring the correct labels' scores to not be the highest. Our property focuses on giving a robust neighborhood against a type of untargeted attacks; Such attacks that aim to bring a specific correct label's score to not be in the top picked labels. Related work:

- Song et al. [7] Introduced targeted white-box attacks for multi-label classification. They approached the problem by formulating it as an optimization problem and using gradient descent to solve it. Through experimentation, they discovered that they could manipulate a multi-label classifier into producing any set of labels for a given input by adding an adversarial perturbation.

- Zhou et al. [33] Suggested generating L_∞-norm adversarial perturbations to trick multi-label classifiers. They solved the problem by transforming the optimization problem of finding adversarial perturbations into a linear programming problem, which can be solved efficiently.
- Another study by Yang et al. [31] Explored the potential for misclassification risk in multi-label classifiers, particularly in worst-case scenarios. They approached the problem by formulating it as a bi-level set function optimization problem, and used random greedy search to find an approximate solution.

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 [16, 12, 13, 17], we use similar approach as MaRVeL's [12].

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] Qingquan Song; Haifeng Jin; Xiao Huang; Xia Hu. Multi-label adversarial perturbations. *IEEE*, 2018.

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

[9] Gulwani S. Seshia S.A. Tiwari A. Jha, S. Oracle-guided component-based program synthesis. *ICSE*, 2010.

[10] Gulwani S. Seshia S.A. Tiwari A.Sara Sabour Nicholas Frosst Geoffrey E. Hinton Jha, S. Dynamic routing between capsules. *NeurIPS*, 2017.

[11] Neil Gong Jinyuan Jia, Wenjie Qu. Multiguard: Provably robust multi-label classification against adversarial examples. *Advances in Neural Information Processing Systems*, 2022.

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

[13] Anan Kabaha and Dana Drachsler-Cohen. Boosting robustness verification of semantic feature neighborhoods. In Gagandeep Singh and Caterina Urban, editors, *Static Analysis*, pages 299–324, Cham, 2022. Springer Nature Switzerland.

[14] 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.

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

[16] Changjiang Li, Shouling Ji, Haiqin Weng, Bo Li, Jie Shi, Raheem Beyah, Shanqing Guo, Zonghui Wang, and Ting Wang. Towards certifying the asymmetric robustness for neural networks: Quantification and applications. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3987–4001, 2022.

[17] Chen Liu, Ryota Tomioka, and Volkan Cevher. On certifying non-uniform bounds against adversarial attacks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4072–4081. PMLR, 09–15 Jun 2019.

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

[19] Kilian Weinberger Minmin Chen, Alice Zheng. Fast image tagging. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[20] Poggio T Papageorgiou, C. A trainable system for object detection. *International Journal of Computer Vision*, 2000.

[21] Bernhard Schölkopf Rohit Babbar. Adversarial extreme multi-label classification, 2018.

[22] 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.

[23] Xiaojiang Peng Jianfei Yang Zhaoyang Zeng Yu Qiao Sijie Ji, Kai Wang. Multiple transfer learning and multi-label balanced training strategies for facial au detection in the wild. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

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

[25] Angelo Sotgiu Ambra Demontis Battista Biggio Marco Gori Stefano Melacci, Gabriele Ciravegna. Domain knowledge alleviates adversarial attacks in multi-label classifiers. *IEEE*, 2022.

[26] Bernt Schiele Sukrut Rao, David Stutz. Adversarial training against location-optimized adversarial patches. *ECCV*, 2020.

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

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

[29] 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.

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

[31] Zhuo Yang, Yufei Han, and Xiangliang Zhang. Characterizing the evasion attackability of multi-label classifiers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10647–10655, May 2021.

[32] 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.

[33] Nan Zhou, Wenjian Luo, Xin Lin, Peilan Xu, and Zhenya Zhang. Generating multi-label adversarial examples by linear programming. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.