Analyzing Neural Network Robustness Using Graph Curvature

Shuhang Tan
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY, USA
tans5@rpi.edu

Jayson Sia Paul Bogdan
Dept. of Electrical and Computer Engineering
University of Southern California
Los Angeles, CA, USA
{jsia, pbogdan}@usc.edu

Radoslav Ivanov
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY, USA
ivanor@rpi.edu

***Abstract*—This paper presents a new look at the neural network (NN) robustness problem, from the point of view of graph theory analysis, specifically graph curvature. Graph curvature (e.g., Ricci curvature) has been used to analyze system dynamics and identify bottlenecks in many domains, including road traffic analysis and internet routing. We define the notion of neural Ricci curvature and use it to identify bottleneck NN edges that are heavily used to "transport data" to the NN outputs. We provide an evaluation on MNIST that illustrates that such edges indeed occur more frequently for inputs where NNs are less robust. These results will serve as the basis for an alternative method of robust training, by minimizing the number of bottleneck edges.**

I. Introduction

Autonomous systems (AS) increasingly use neural networks (NNs) due to their ability to process high-dimensional data such as camera images [1], LiDAR scans [2] and textual prompts [3]. At the same time, NNs are known to suffer from robustness vulnerabilities: a slightly perturbed or out-of-distribution input [4], [5] may lead to very different and unexpected outputs. In turn, such vulnerabilities may severely compromise the safety and predictability of NN-based AS.

Since the discovery of NN robustness issues [4], there has been an impressive amount of research on this topic. Researchers have developed a number of robust training methods, including adversarial training [6], certified robustness [7], [8], knowledge distillation [9], and semi-infinite constrained learning [10]. Although significant progress has been made, training robust NNs remains largely an unsolved and very challenging problem (e.g., the current leader on the CIFAR-10 robustness leaderboard [11] can only achieve high robust accuracy for perturbations of at most 8/255).

We note that the vast majority of existing methods approach the problem from an optimization point of view: e.g., in adversarial training the goal is to train a NN that minimizes the loss not only on training data but also on the worst-case bounded perturbations of that data. This bilevel non-convex optimization problem is challenging to solve and leads to suboptimal solutions, especially if gradient descent is used.

We take a fresh look at NN robustness through the lens of graph theory and network science analysis, in particular graph curvature (GC). GC (e.g., Ricci curvature [12]) has been effectively applied in numerous domains that can be modeled as graphs, including road traffic analysis [13], [14], internet routing [15], machine learning [16], [17], and biological networks [16], [18], due to its ability to capture intrinsic geometric and local structure of the space, such as connectivity and robustness in networks. GC can quantify the importance of specific edges; for example, an edge with *negative curvature* can be considered a bottleneck and is greatly important for the overall graph functionality, e.g., such an edge may connect different communities within the graph [19], [20].

In this paper, we employ GC in order to analyze the robustness of NN classifiers. We introduce the notion of neural Ricci curvature (NRC) that captures the bottleneck intuition of standard Ricci curvature – if an edge has a negative NRC, then it is heavily used by the NN and is thus likely a source of robustness vulnerability. To calculate the NRC, we construct a neural data graph, i.e., a graph in the shape of the NN architecture, where edges are weighted by a combination of the NN weights and the magnitude of data that goes through each edge when an example is provided as input.

We evaluate the significance of the NRC using NNs trained on MNIST. We show that neural data graphs corresponding to more robust examples (i.e., examples which are correctly classified even for an adversarial perturbation) indeed have fewer negative-NRC edges. The results are consistent across architectures, including adversarially trained ones. This result will serve as the basis for an alternative, graph-based, method for robust training, that minimizes the number of negative-NRC edges and promotes balanced usage of all NN edges.

In summary, this paper makes two contributions: 1) we define the concepts of *neural data graphs* and *neural Ricci curvature* that can be used to identify bottleneck NN edges that contribute to robustness issues; 2) we provide an evaluation on MNIST that demonstrates that bottleneck edges indeed occur more frequently in examples where NNs are less robust.

II. Background

The concept of Ricci curvature [21] is used in Riemannian geometry to quantify the degree to which the geometry of a

This material is based upon work supported by the National Science Foundation (NSF) under Award No. 2403616, as part of the NSF Cyber-Physical Systems Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

space deviates from being flat, as is the case for Euclidean space. In continuous manifolds, positive curvature is seen in spherical surfaces where geodesics converge, negative curvature is found in hyperbolic surfaces where geodesics diverge, and zero curvature characterizes flat Euclidean surfaces with parallel geodesics. The Ollivier-Ricci curvature (ORC) [12] serves as a discrete analogue, e.g., in the case of graphs, to curvature measurements in continuous spaces, and it is computed using transport theory via the Wasserstein distance.

Definition 1 (Ollivier-Ricci Curvature [12])**.** *Given a graph $G(V, E)$ with vertex set V and edge set E, the ORC $\kappa(u, v)$ between two adjacent vertices u and v is given by*

$$\kappa(u,v) = 1 - \frac{W_1(m_u, m_v)}{d(u,v)}, \tag{1}$$

where $d(u, v)$ is the shortest-path distance between u and v, m_u and m_v are the probability distributions over the neighbors of u and v, respectively; $W_1(m_u, m_v)$ is the Wasserstein distance between distributions m_u and m_v and is given by

$$W_1(m_u, m_v) = \inf_{\mu_{u,v} \in \Pi_{u,v}} \sum_{(u',v') \in V \times V} d(u', v') \mu_{u,v}(u', v'),$$

where $\Pi_{u,v}$ is the set of probability measures $\mu_{u,v}$ that capture all possible ways of transferring mass from m_u to m_v, i.e.,

$$\sum_{v' \in V} \mu_{u,v}(u', v') = m_u(u'), \quad \sum_{u' \in V} \mu_{u,v}(u', v') = m_v(v').$$

For unweighted nodes, the probability distribution is typically distributed uniformly to all the neighbors of u and v. Otherwise, the probability distribution is adjusted based on the edge weights, e.g., using an exponential function [22].

A positive curvature indicates that nodes within a particular region are tightly connected (due to a smaller W_1), suggesting a robust community structure. A negative curvature points to areas with more dispersed connections, often signifying edges that act as bridges between different communities. For example, if an edge e between u and v is a bottleneck, then all paths from u's neighbors to v's neighbors have to go through e; in this case, $\kappa(u, v) < 0$ since $W_1 > d(u, v)$. This makes ORC particularly effective at identifying bottleneck edges that may introduce robustness issues, i.e., the graph functionality is not robust to removing such negative-curvature edges.

III. Problem Statement

Consider a trained NN classifier $f_\theta : \mathcal{X} \to \{1, \dots, N\}$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of all input examples, and each example is assigned a label from 1 to N. We assume f_θ is an L-layer feedforward NN (possibly including convolutional layer), parameterized by $\theta = \{(W_1, b_1), \dots, (W_L, b_L)\}$, where W_i and b_i are the weights and biases of layer i. Thus, each layer i can be written as a function $f_i(x) = \sigma(W_i x + b_i)$, where σ is the ReLU activation: $\sigma(x) = \max\{0, x\}$.

An example (x, y) is *ε-robust* if there exists no ε-bounded perturbation of x that changes the label predicted by f, i.e.,

$$f_\theta(x) = f_\theta(x + \delta) = y, \forall \delta \in \mathbb{R}^n \; \|\delta\|_\infty \leq \varepsilon, \tag{2}$$

where y is the true (unknown) label of x. An example is called *ε-nonrobust* if there exists a δ that satisfies $\|\delta\|_\infty \leq \varepsilon$ for which $f_\theta(x) \neq f_\theta(x + \delta)$. Robust test accuracy is the fraction of ε-robust examples in the test set. Although NN robustness is distinct from graph robustness, we hypothesize that non-robust NNs correspond to non-robust graphs.

The problem considered in this paper is to define the notions of a neural data graph $G_{f_\theta,x}$ and a corresponding neural Ricci curvature. In particular, the NRC definition must capture the notion of a bottleneck edge such that a NN with more negative-NRC edges is less robust to input perturbations.

Notation. We use $n_{l,i}$ to denote neuron i in layer l in the NN; $n_{l,i}(x)$ denotes the neuron's activation given NN input x. Similarly, $w_{l,jk}$ denotes the NN weight on the edge between $n_{l,j}$ and $n_{l+1,k}$, i.e., matrix entry $[W_l]_{kj}$. Given graph nodes u and v, we use $w(u, v)$ to denote the weight of edge (u, v).

IV. Neural Ricci Curvature

Our approach is motivated by prior work on analyzing the robustness of road and transit traffic networks to individual segments [13], [14], [16]. To model the problem as a graph, one can map each intersection to a node and each segment to an edge; the edge weight is determined by historical data, e.g., travel time per passenger. Thus, if an edge has a negative ORC, i.e., its cost is lower than the costs of paths through the node's neighbors, then this edge is likely a bottleneck.

The above example has obvious analogues and differences with respect to NNs. The main similarities are: 1) a NN has a natural graph structure; 2) NN edges can be considered as roads that carry data; 3) NN weights can be thought of as edge weights such that a larger NN weight means more data gets through (i.e., less travel time). However, two NN characteristics distinguish it from the traffic network case: i) the NN has non-linear activations which break the traffic analogy (different levels of traffic "get through" depending on the input); ii) the NN has positive and negative weights, which makes it impossible to apply the vanilla ORC analysis.

In what follows, we define the concepts of a neural data graph and neural Ricci curvature, through iteratively addressing the challenges above: 1) we start with a neural graph; 2) we address the non-linearity challenge through calculating each ReLU's phase (0 or linear) for the current input x; 3) we address the mixed-sign challenge through normalizing the NN weights for the current input x.

Definition 2 (Neural Graph)**.** *Given a NN f_θ, we define a neural graph G_{f_θ} as follows:*

- *each NN neuron becomes a node in G_{f_θ} and each NN edge becomes an edge in G_{f_θ};*
- *each edge $(n_{l,i}, n_{l+1,j})$ is assigned weight $1/|w_{l,ij}|$.*

Definition 2 is based purely on the NN and thus cannot address the challenges mentioned above. The following construction provides an approach that alleviates those challenges.

Definition 3 (Neural Data Graph)**.** *Given a NN f_θ and an example x, we define a neural data graph $G_{f_\theta,x}$ as follows:*

- *construct neural graph G_{f_θ} according to Definition 2;*
- *an edge $(n_{l,i}, n_{l+1,j})$ is removed from G_{f_θ} if $n_{l,i}(x) = 0$, i.e., the ReLU is in its zero phase;*

Algorithm 1 Mixed-Sign Weights Normalization

```
Input: Mixed-sign NN weights $w_{l,1j}, \dots, w_{l,Kj}$, neuron activations $n_{l,1}(x), \dots, n_{l,K}(x)$
    //Assuming layer $l$ has $K$ neurons
Output: Graph weights $w(n_{l,1}, n_{l+1,j}), \dots, w(n_{l,K}, n_{l+1,j})$
sum = $\sum_{i=1}^{K} w_{l,ij} n_{l,i}(x)$
//it must be the case that $sum \geq 0$; otherwise ReLU would be in 0 phase
for $i = 1$ to $K$ do
    if $w_{l,ij} < 0$ then
        $w(n_{l,i}, n_{l+1,j}) = 0$
    else
        $pos\_sum = \sum_{i=1}^{K} 1_{w_{l,1j}>0} w_{l,1j} n_{l,1}(x)$
        $\hat{w}_{l,ij} = w_{l,ij} \frac{sum}{pos\_sum}$
        $w(n_{l,i}, n_{l+1,j}) = \frac{1}{\hat{w}_{l,ij}}$
    end if
end for
```

- *if weights $w_{l,1j}, \dots, w_{l,Kj}$ (where K is the number of neurons in layer l) going into node $n_{l+1,j}(x) > 0$ have mixed signs, normalize weights using Algorithm 1.*

The last two parts of Definition 3 modify the original neural graph G_{f_θ} based on the contribution of the input example x. Algorithm 1 only applies when $n_{l_1,j}(x) > 0$ since otherwise the ReLU would be its zero phase and all outgoing edges would be removed. Algorithm 1 normalizes the weights such that negative weights are reset to 0 and positive weights are normalized so that overall sum remains the same. Given Definition 3, we are ready to define the neural Ricci curvature, which is essentially the ORC applied to the neural data graph.

Definition 4 (Neural Ricci Curvature)**.** *Consider a NN f_θ, an input example x and a corresponding neural data graph $G_{f_\theta,x}$, as defined in Definition 3. The NRC of a NN edge $(n_{l,i}, n_{l+1,j})$ is defined as the ORC of the corresponding edge in $G_{f_\theta,x}$.*

V. Evaluation

The evaluation aims to demonstrate the following: given a NN, ε-robust examples with higher ε tend to result in neural data graphs with fewer negative-NRC edges. Note that this result is orthogonal to the overall NN robustness – even non-robust NNs exhibit some ε-robust examples with large ε. This finding implies that training NNs with fewer negative-NRC edges would be an effective method for robust training.

We evaluate the NRC concept on the MNIST dataset [23]. MNIST consists of 28×28 grayscale images of handwritten digits; there are 60,000 training images and 10,000 test images. We use two fully-connected NN architectures, [15,20] and [15,25,20,15], where notation $[K_1, \dots, K_L]$ means the NN has L layers, with K_i neurons in layer i. Each architecture is trained in three ways: 1) using cross-entropy loss; 2) using cross-entropy with weight decay regularization; 3) using adversarial training [6]. For further evaluation, we also include a convolutional NN (CNN), trained with cross-entropy, with two convolutional layers (first layer has six 6×6 kernels, stride of two; second layer has 16 6×6 kernels, stride of two) and two fully-connected layers, [120,84]. The models' robust accuracies are reported in Table I. All experiments were run on a 95-core machine with an NVIDIA A40 unit; calculating curvatures per example took on average 1.2s, 1.4s and 3.2s for the two-layer NN, four-layer NN and CNN, respectively.

NN setup	$\varepsilon = 0.03$	$\varepsilon = 0.07$	$\varepsilon = 0.1$	$\varepsilon = 0.2$
[15,20], CE	0.517	0.044	0.005	0.000
[15,20], WD	0.851	0.421	0.175	0.014
[15,20], AT	0.845	0.766	0.685	0.270
[15,25,20,15], CE	0.471	0.054	0.007	0.000
[15,25,20,15], WD	0.801	0.311	0.109	0.001
[15,25,20,15], AT	0.862	0.780	0.692	0.253
CNN, CE	0.939	0.725	0.409	0.017

TABLE I: Robust test accuracy (evaluated using the projected gradient descent attack [6]) for all setups: cross-entropy (CE), cross-entropy + weight decay (WD), adversarial training (AT).

NN setup	$\varepsilon = 0.03$	$\varepsilon = 0.07$	$\varepsilon = 0.1$	$\varepsilon = 0.2$
[15,20], CE	1.48	1.30	1.28	N/A
[15,20], WD	1.47	1.45	1.37	1.24
[15,20], AT	1.15	1.15	1.14	1.12
[15,25,20,15], CE	2.04	1.81	1.66	N/A
[15,25,20,15], WD	1.93	1.92	1.86	N/A
[15,25,20,15], AT	2.10	2.09	2.10	2.08
CNN, CE	3.43	3.42	3.44	3.31

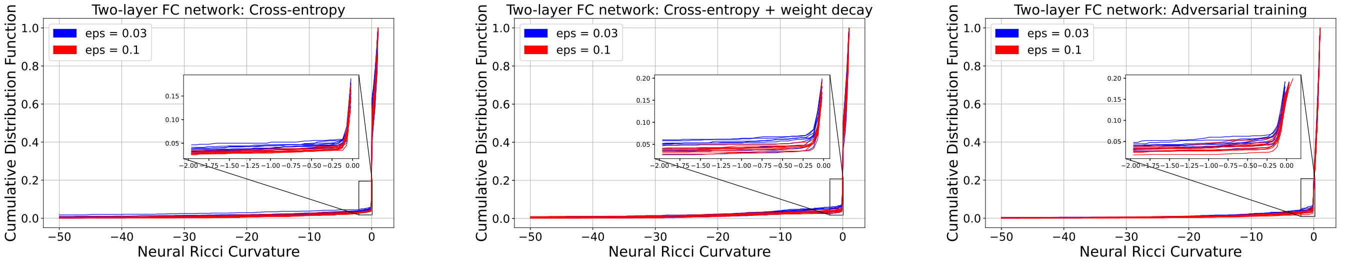
TABLE II: Average AUC over all labels for all considered setups: cross-entropy (CE), cross-entropy + weight decay (WD), adversarial training (AT).

To perform the evaluation, we use the test set to identify ε-robust and ε-nonrobust images for each setup, where ε is incremented from 0.03 to 0.2. We calculate the NRCs for each resulting neural data graph and plot them using a cumulative distribution function (CDF). As shown in Fig. 1, the CDF grows faster for examples which are less ε-robust, i.e., those examples have more negatve-NRC edges, on average, than the graphs corresponding to more ε-robust examples.

For further evaluation, we also present the average area under the CDF curve (AUC) per label (averaged over 70 test examples). Fig. 2 provides the average results for the two-layer fully-connected (FC) NN. Fig. 2a very clearly demonstrates that the AUC decreases with larger ε, i.e., more ε-robust examples result in fewer negative-NRC edges on average. Although the benefit is less pronounced for the other setups, there is still a clear downwards trend as ε is increased. Finally, Table II presents the AUC results for all setups. Once again, we emphasize that the same trend can be observed across all considered setups. Finally, we note that the CNN trends are less emphasized, which is likely due to the fact that convolutional layers have very few edges per node; we will explore this phenomenon at greater depth in future work.

VI. Conclusion

This paper introduced neural data graphs and neural Ricci curvature, which provide an alternative way of analyzing NN robustness. We presented an evaluation on the MNIST dataset to demonstrate that more robust NNs (as well as more ε-robust examples) result in fewer negative-NRC edges. In future work, we will perform an evaluation over multiple datasets

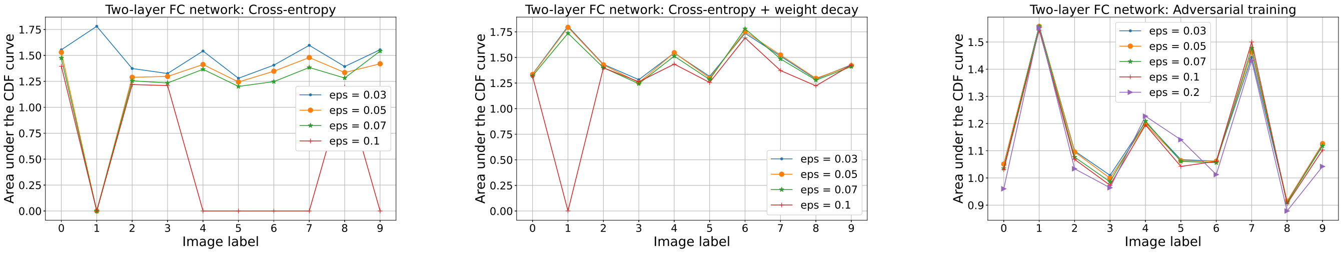


(a) Cross-entropy, label 2.

(b) Cross-entropy + weight decay, label 4.

(c) Adversarial training, label 3.

Fig. 1: CDF plots of 10 test examples each for two levels of ε, for the three two-layer NN setups. Note that the 0.03-robust examples are chosen such that they are 0.05-nonrobust.



(a) Cross-entropy.

(b) Cross-entropy + weight decay.

(c) Adversarial training.

Fig. 2: AUC results, averaged over 70 test examples per label and per ε, for the three two-layer NN setups. Values of 0 mean that no robust examples could be found for that value of ε.

and robust training methods [24]–[26], and will explore an approach for robust training, e.g., by regularizing training through a term that penalizes low-curvature edges.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Case study: verifying the safety of an autonomous racing car with a neural network controller," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020.

[3] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv:1312.6199*, 2013.

[5] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *ICML*, 2019.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv:1706.06083*, 2017.

[7] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *international conference on machine learning*. PMLR, 2019, pp. 1310–1320.

[8] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International conference on machine learning*. PMLR, 2018, pp. 5286–5295.

[9] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE symposium on security and privacy (SP)*, 2016, pp. 582–597.

[10] A. Robey, L. Chamon, G. J. Pappas, H. Hassani, and A. Ribeiro, "Adversarial robustness with semi-infinite constrained learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6198–6215, 2021.

[11] "RobustBench: A standardized benchmark for adversarial robustness," https://robustbench.github.io/.

[12] Y. Ollivier, "Ricci curvature of Markov chains on metric spaces," *J. Functional Analysis*, vol. 256, no. 3, pp. 810–864, 2009.

[13] Y. Wang, Z. Huang, G. Yin, H. Li, L. Yang, Y. Su, Y. Liu, and X. Shan, "Applying ollivier-ricci curvature to indicate the mismatch of travel demand and supply in urban transit network," *International Journal of Applied Earth Observation and Geoinformation*, vol. 106, 2022.

[14] L. Gao, X. Liu, Y. Liu, P. Wang, M. Deng, Q. Zhu, and H. Li, "Measuring road network topology vulnerability by ricci curvature," *Physica A: Statistical Mechanics and Its Applications*, vol. 527, 2019.

[15] C.-C. Ni, Y.-Y. Lin, J. Gao, X. D. Gu, and E. Saucan, "Ricci curvature of the internet topology," in *2015 IEEE conference on computer communications (INFOCOM)*. IEEE, 2015, pp. 2758–2766.

[16] M. R. Znaidi, J. Sia, S. Ronquist, I. Rajapakse, E. Jonckheere, and P. Bogdan, "A unified approach of detecting phase transition in time-varying complex networks," *Scientific reports*, vol. 13, no. 1, 2023.

[17] H. Li, J. Cao, J. Zhu, Y. Liu, Q. Zhu, and G. Wu, "Curvature graph neural network," *Information Sciences*, vol. 592, pp. 50–66, 2022.

[18] H. Farooq, Y. Chen, T. T. Georgiou, A. Tannenbaum, and C. Lenglet, "Network curvature as a hallmark of brain structural connectivity," *Nature communications*, vol. 10, no. 1, p. 4937, 2019.

[19] J. Sia, E. Jonckheere, and P. Bogdan, "Ollivier-Ricci curvature-based method to community detection in complex networks," *Scientific Reports*, vol. 9, no. 1, p. 9800, 12 2019.

[20] J. Sia, W. Zhang, E. Jonckheere, D. Cook, and P. Bogdan, "Inferring functional communities from partially observed biological networks exploiting geometric topology and side information," *Scientific Reports*, vol. 12, no. 1, p. 10883, 2022.

[21] S. Bochner, "Vector fields and ricci curvature," 1946.

[22] C.-C. Ni, Y.-Y. Lin, F. Luo, and J. Gao, "Community detection on networks with ricci flow," *Scientific reports*, vol. 9, no. 1, p. 9984, 2019.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] F. Brau, G. Rossolini, A. Biondi, and G. Buttazzo, "Robust-by-design classification via unitary-gradient neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023.

[25] K. Leino, Z. Wang, and M. Fredrikson, "Globally-robust neural networks," in *Intl. Conf. on Machine Learning*, 2021, pp. 6212–6222.

[26] D. Zou, R. Balan, and M. Singh, "On lipschitz bounds of general convolutional neural networks," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1738–1759, 2019.