Harmonic Machine Learning Models are Robust

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce Harmonic Robustness, a powerful and intuitive method to test the robustness of any machine-learning model either during training or in black-box real-time inference monitoring without ground-truth labels. It is based on functional deviation from the harmonic mean-value property, indicating instability and lack of explainability. We show implementation examples in low-dimensional trees and feedforward NNs, where the method reliably identifies overfitting, as well as in more complex high-dimensional models such as ResNet-50 and Vision Transformer where it efficiently measures adversarial vulnerability across image classes.

1 Motivation and Introduction

Modern application of Machine Learning (ML) across all industries faces numerous challenges in maintaining quality of predictions: from the training phase where one must choose the “best" model within a sea of architectures and hyperparameters to maximize performance without overfitting the training data, while balancing with explainability and fairness in the context of Responsible AI; to the inference phase where, in the face of production latency and throughput constraints, one must efficiently monitor for performance degradation due to data drift; ideally this latter triggers the model re-training phase, where one must revisit the model with freshly-labelled data, which, however, in many applications such as credit card fraud may not be available till after a significant time lapse, sometimes months later.

To address these challenges, we propose a simple geometric technique which enjoys several mitigating properties:

- It is model-agnostic (black-box) and unsupervised, i.e., requiring no knowledge of model inner workings, ground-truth labels or other auxiliary data.
- Its computation is algorithmically simple, linear in the number of data points tested, and has good statistical sampling convergence.
- It reliably measures relative overfitting between two models on the same training data.
- It precisely measures model robustness across feature space and can immediately indicate data drift in online monitoring.
- It is indicative of model explainability.

The particular proposal is to measure the “harmoniticity" of the model, specifically the degree to which the model function f satisfies the harmonic property,

$$\nabla^2 f = 0 \tag{1}$$

Functions which satisfy (1), i.e, “harmonic functions”, occur frequently in physics as solutions to equilibrium problems involving minimization of energy, e.g. soap bubbles stretched on a boundary, electrostatic field configurations, and heat flow (see Fig 1). They form smooth, minimal interpolations

between boundary values, and most importantly to our present discussion exhibit the “mean-value property",

$$f(x) = \frac{1}{S} \int_{B(x)} f d\Omega \tag{2}$$

which, in plain English, says that the value of the function at any point is the surface average of the function over a ball of any radius surrounding the point (incidentally, (1) and (2) are equivalent definitions)[1]. The metric we propose, “anharmoniticity" or γ for brevity, measures how well (2) is satisfied over feature space, computing the difference between the function and its ball-averaged value:

$$\gamma(x) \equiv |f(x) - \frac{1}{S_{r,n}} \int_{B(x,r)} f d\Omega_{r,n}| \tag{3}$$

where now we explicitly introduce the parameters r (ball radius) and n (feature dimension) as the implementation details will depend on these. As the behavior of $f(x)$ may vary wildly over feature space, so will $\gamma(x)$ depending on the degree to which f behaves in accordance with (2) for some reasonable fixed choice of r. By association this will indicate which regions of feature space are “more harmonic" for this model, and the average value of $\gamma(x)$ over feature space provides a summary “anharmoniticity" metric.

At this point, we can verify the above claimed mitigating properties of this metric:

- Measuring γ as per (3) requires nothing more than black-box access to the model, as is expedient in an inference setting; this also allows testing of closed-source models without having to request access to model details, facilitating efficiency of testing and helping to keep the industry honest.
- As γ only requires computing the average value of f at a number of data points approximating a ball, this is linear in the number of points and amenable to sampling.
- γ is proportional to the complexity of the decision surface; in particular for a binary classifier it is proportional to the length of the decision boundary, which is positively correlated with overfitting (see Appendix).
- If γ changes over time in online events, there must be data drift; this is great for online monitoring where it is paramount to raise alerts as soon as a performance issue arises. If production monitoring shows an increase in γ, one can pinpoint the data points responsible and investigate that region of feature space more fully for counterfactuals — this might trigger the need for re-training with more data or modeling in that region.
- Harmonic functions are natively explainable since, by the mean-value property, the ’explanation’ of any point is that it is the average of the points around it, which in turn are ‘explained’ by their neighboring points, etc., all the way up to the feature boundaries which have values fixed by some standard. The premier example is the linear function, of course trivially harmonic by (1) and explainable by direct proportionality. Thus, the closer γ is to zero, the more explainable the model will be. Conversely, the more a model fails (2) at some point the more difficult it may be to explain, e.g., in the fraud domain if the average of several non-fraudulent events was predicted to be fraudulent.

In short, γ is a proxy for the measure of model robustness in stability of prediction, resilience to data drift, and ease of explainability.

Note, however, that the aim of this programme is certainly not to have a model be completely harmonic — indeed by (2) it is easy to check that pure harmonic functions can have no local minima or maxima, and that is too restrictive for real-world models.

Yet, we believe a robust, explainable ML model should be at least locally close to harmonic in *most* of feature space[2], especially for production applications where stability is business-critical.

[1] An easy informal way to see this equivalence is to recognize (1) as the divergence of the gradient ($\nabla^2 f = \bigtriangledown \cdot \bigtriangledown f$): the gradient expressing the change of f in all directions, if its divergence is zero then ’change in f’ neither flows into or out of any given point, hence the average change of f on any ball around that point is 0, relative to its value at the point.

[2] This can be made precise by choosing a value of r that is ’small’ relative to typical distance between data points. Though as we’ll see below, the exact choice is not critical.

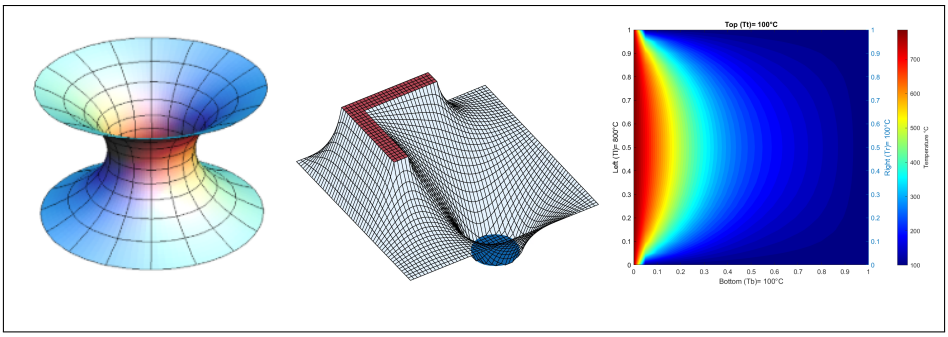

Figure 1: Examples of harmonic functions that appear in Nature: soap films[1], electrostatic potentials[2], and heat flows[3].

Harmonic functions exhibit the minimal curvature necessary to interpolate between fixed data points (see Appendix), and hence satisfy a certain Occam's Razor of machine learning. Most real-world models will of course deviate from pure harmoniticity, but γ gives us a way to track and quantify the deviation.

Finally, as far as we know, this is the first instance in the literature of an overarching, limiting *algebraic* standard on a ML function for the purpose of quality and stability. Our choice of the harmonic property is not in any way sacrosanct, but it is intuitive and accessible for quick and direct testing via the mean-value property (2), giving correlation with stability and ease of interpretation. It may be that some other class of functional algebraic constraints also captures this and more, a topic we leave open to the community to explore.

2 Related Work

Techniques to measure goodness of a predictive model are of course as old as the field of Machine Learning itself, traditionally centered on time-tested metrics such as precision, recall, F-score, AUC, etc. where the ground-truth labels of a test set are known. On the other hand, we are chiefly concerned with the real-world problem of measuring model robustness without access to ground truth labels, as for example occurs in a purely online inference environment with only black-box access to the model in conjunction with a live data stream. For this is the real, minimal environment in which most practioners and end users of ML operate. Statistical techniques such as outlier or anomaly-detection [4] and distributional shift [5] enjoy usage here to give important hints of, but not true indications of, model robustness. Gradient-based methods such as PDP and ICE [6] where one looks for sudden changes in the decision function over feature space likewise may provide hints of robustness changes, though this differs from the current proposal as γ is measuring more than just the local feature sensitivity in the function, which may in fact be proper and desired behavior for, e.g., a steep linear response; rather, γ is measuring departure from *explainable* sensitivity as one sees in harmonic functions obeying the mean-value property (2).

Existing work [7] as well as a recent survey [8] reviews 23 metrics which are useful in the online inference setting, e.g., Average Confidence of True Class (ACTC) and Noise Tolerance Estimation (NTE). Within the black-box setting the metrics are essentially measuring how readily the predicted class label changes across feature space either due to targeted gradient-based search or random perturbation. These fit in the realm of Adversarial Machine Learning [9] which has blossomed into its own subfield, quite rightly dedicated to understanding the vulnerability of popular ML models to attacks [10][11] based on perturbing input data points. Recent work has found that adversarial weakness becomes more prevalent with increasing number of feature dimensions [12][13], and sensitive data domain dependence arises; unsurprisingly this is most apparent in image classification tasks [14], the premier testing-ground of adversarial ML, as each data point can easily contain thousands to millions of features (pixels); it is important to note, however, that many other domains, e.g., financial modeling, can contain thousands or more features and likewise be highly vulnerable to attacks [15]. Adversarial analyses also typically focus on class label changes, differing from our metric which is more precisely measuring the numerical stability of the prediction (logit) according

to the standard of harmonic geometry, and not merely the crossing of such predictions over discrete thresholds leading to class label changes.

There have for a number of years been works focusing on the relationship between stability and geometry of the classifier [16][17][18]: what these works find is that there is a correlation between adversarial weakness and curvature of the decision surface. This has even fueled investigation into a new way of classification using the average or majority-vote in a hypercube neighborhood of each point [19]. This is corroborated by the present study as well, for harmonic functions describe minimal surfaces with constant mean curvature [20], hence should have minimal adversarial weakness.

3 Method

Computing $\gamma(x)$ for a model as per its mathematical definition (3) is an extremely simple and straightforward procedure, which we detail below in pseudocode:

Algorithm 1 Computation of γ at a point x in feature space

procedure $\gamma(x)$
    ballPoints $\leftarrow \mathrm{Ball}(x, r)$
    N $\leftarrow size$(ballPoints)
    ballValue $\leftarrow 0$
    for `each` point `in` ballPoints **do**
        ballValue $+=$ f(point)
    end for
    ballAvg $\leftarrow$ ballValue/N
    return $|f(x) -$ ballAvg$|$
end procedure

One can then average $\gamma(x)$ over a region of feature space to get γ for that region, for example the convex hull of a training set or all feature points seen in some inference production window.

In the above algorithm, the primary consideration is how to get the ball of radius r around the point x, i.e., $\mathrm{Ball}(x, r)$, remembering that in general x is a vector in some possibly high number of dimensions. For any digital computation we will of course have to approximate a continuous ball with a discrete number of points. The easiest-to-code solution is to construct a large number N of random vectors around x, each normalized to some small magnitude r, hoping for isotropy and centrality (zero overall vector sum). The random-walk behavior of N random vectors will however doom one to a $\sqrt{N}$ bias in one direction or another (see Appendix).

A better solution from a theoretical perspective is to form the "n-simplex" around each point. In two dimensions, for example, the 2-simplex is an equilateral triangle; in three dimensions a tetrahedron, etc. (see Appendix). In any arbitrary number of dimensions, the n-simplex centered about a point will be maximally symmetric, hence ideally space-covering. One can further add balanced rotations of the basic n-simplex, the more of which you add the closer the discrete approximation converges to the continuous ball.

With a bit of linear algebra to compute the n-simplices (see Appendix), numerical trials indeed show that n-simplices symmetrically cover space much more effectively than random vectors and, when used to approximate the ball in our algorithm, accurately identify pure harmonic versus non-harmonic functions, the details of which the interested reader may refer to in the Appendix. What we would like to focus on in the remainder of this paper is actual ML models, as readers will find this most applicable to their work.

4 Application to low-dimension models

For clarity in demonstrating the method let us first focus on basic models targeting a small, well-understood dataset in the ML community: the Wine dataset [21], describing 13 features of three different wines grown in the same region of Italy. We further restrict this analysis to just two of those dimensions, “flavanoids" and “OD280/OD315 of diluted wines", in order to show a model taking a two-dimensional input vector to a scalar output (the wine class label). What we seek to show is how

computing γ on models trained with these features on this data indicate the property of being well-fit or overfit, hence vulnerable to adversarial attacks or under-performance in production, purely from inference on out-of-training points without referencing any ground-truth labels.

We split the original data 80/20 to a Train/Test set, foregoing the usual split to a Validation set not just due to data sparsity (there only being around 100 data points in this set), but to show how the present technique can by itself indicate overfitting. We train four models: two Gradient Boosted Decision Trees (GBDT), "GBDT-1", with hyperparameters optimized on a grid search with 10-fold cross-validation, and the second, "GBDT-2", chosen with more extreme hyperparameter values; and two feedforward neural nets (multi-layer perceptron), a 1-hidden-layer model "MLP-1" and a much more parameterized 3-layer model "MLP-2" likely to overfit. Hyperparameters for all models are shown in Tables 1 and 2.

Intuition should tell us that the over-parameterized and under-regularized models GBDT-2 and MLP-2 will perform better than GBDT-1 and MLP-1 on the Train set but not so on the Test set, and this is indeed the case as shown in the tables below. What we will show is this could also have been gleaned from the shapes of the decision boundaries for these classifiers, shown in the top row of Fig. 2: note how the shape of the overfit models' decision boundaries is more complicated than that of the well-fit models. Computing $\gamma(x)$ with $r = 0.05$ on a grid[3] in a region safely enclosing all data points, we get the bottom row of plots. Notice that γ is non-zero only around the decision boundary, which for the overfit functions is always longer — these latter will thus have higher average γ, as we confirm in the tables. Computing γ thus identifies potential overfitting without need for checking a labeled validation or test set.

What's also interesting is that both wellfit models (GBDT-1 and MLP-1) have the same Test performance of 83%, but GBDT-1 has a slightly better γ (0.014) versus that of MLP-1 (0.016). One can easily see this difference from the lengths of the decision boundaries: the GBDT boundary consists of nearly straight lines while the MLP boundary is more curved. This illustrates how a model trainer with these test results might, from the perspective of robustness, prefer the GBDT model for use in inference.

Table 1: GBDT models

	GBDT-1	GBDT-2
max_depth	1	100
n_estimators	5	200
min_samples_split	2	2
learning_rate	0.1	1
Train accuracy	85%	100%
Test accuracy	83%	80%
γ (r=0.05)	0.014(2)	0.051(2)

Table 2: MLP models

	MLP-1	MLP-2
max_iter	200	1000
layer dims	(2,100,1)	(2,100,500,1000,1)
initial learning_rate	0.001	0.01
alpha	$1 \cdot 10^{-4}$	0
Train accuracy	82%	86%
Test accuracy	83%	79%
γ (r=0.05)	0.016(1)	0.027(1)

It should be clear from the above, then, that the Harmonic Robustness metric clearly works on simple ML functions in low numbers of dimensions, where we can visually confirm areas of feature space which are more robust and compare robustness of different models over the same space.

5 Application to high-dimension models

In this section we will consider the more complex case of higher-dimensional models to illustrate how the technique adapts.

The challenge of models over a larger number of dimensions (into the thousands or even millions) is three-fold:

1. High-dimensional simplices are more expensive in compute and storage.
2. High-dimensional models are typically more complex and take longer to run.
3. γ itself might be high-dimensional and interpretation is not straightforward.

[3]Dependence on r or grid-size was very mild; see Appendix.

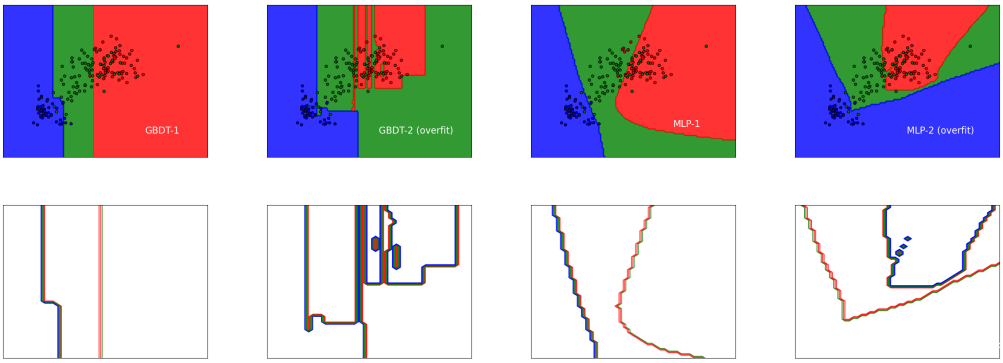

Figure 2: Decision regions (top row) and gamma contours (bottom row) of the two classifiers: GBDT (left), and MLP (right). Gamma contours for radius=0.05 closely follow decision boundaries. Boxed region shown corresponds to [[0,5],[1,4]] in the x ("flavanoids") and y ("OD280/OD315") plane.

For the first challenge, Mathematics is actually kind to us, where it turns out that for large n the n-simplex is approximately the same as the vertices of the n-dimensional hypercube (see Appendix), and that is trivial to compute. For the second challenge, we will have to limit the number of points on the ball in order to be able to compute γ in a reasonable amount of time. Thus we may take a random sampling of the hypercube as a necessary approximation. Finally, if the output is not just a scalar, but rather multidimensional, one must decide whether some additional transformation is needed for interpretation. To take a weather example, if the model output is a 3-dimensional wind velocity, then $\overrightarrow{\gamma}$ represents the instability in velocity, and one might want to take its magnitude or angle with respect to north to interpret as speed instability or directional bias, respectively.

For the purpose of demonstration, we choose here to focus on high-dimensional image-classification models, due to popular practicality and ease of interpretation. The inputs (pixels) and outputs (class logits) are typically both high-dimensional, and would thus serve to illustrate the behavior of any other high-dimensional model as well. In particular, we consider ResNet-50 [22] and the Vision Transformer [23].

5.1 ResNet-50 and ViT

ResNet-50 and Vision Transformer (ViT) are image classifiers trained on 1000 distinct classes. To keep things manageable in this short work, we employ several restrictions: (1) Data is restricted to grayscale images: the value of every pixel is thus an integer from 0 to 255; (2) Images are rescaled to 100x100 resolution: each image will thus be a 10000-dimensional vector; (3) Approximate simplices: 10000-simplices are well-approximated as 1-hot vectors on the 10000-dim unit-hypercube as noted above; we scale each vector to magnitude 100 which amounts to a significant tone change at the position of the corresponding pixel. To increase ball coverage, we will use the simplices together with their reflections (anti-simplices); (4) Random sampling from simplices: rather than compute at all 20000 ball points (10000 simplex + 10000 anti-simplex) for each image, we take a random 0.1% sampling of such; (5) We compute γ only in the logit occupying the dimension of its predicted class label.

We apply these models to an animals test set [24] consisting of over 20k color images of animals in various resolutions from a pre-determined set of 10 classes (dog, horse, elephant, butterfly, chicken, cat, cow, pig, spider, squirrel). Then, as described above, we rescale each image to 100x100 pixels and convert to grayscale before computing γ in its predicted class logit dimension. We evaluate 100 images per class, which will be sufficient to see the trends in robustness and justify the use of our approximations above.

For each image, we also execute an adversarial search process wherein we follow the gradient of γ for 25 iterations, recording the final image and its predicted class (see Algorithm 2 below). Each such image being only 25 pixels disparate from its original form, a change in class label is interpreted as "instability" in the original image, reminiscent of earlier gradient-based adversarial work [25].

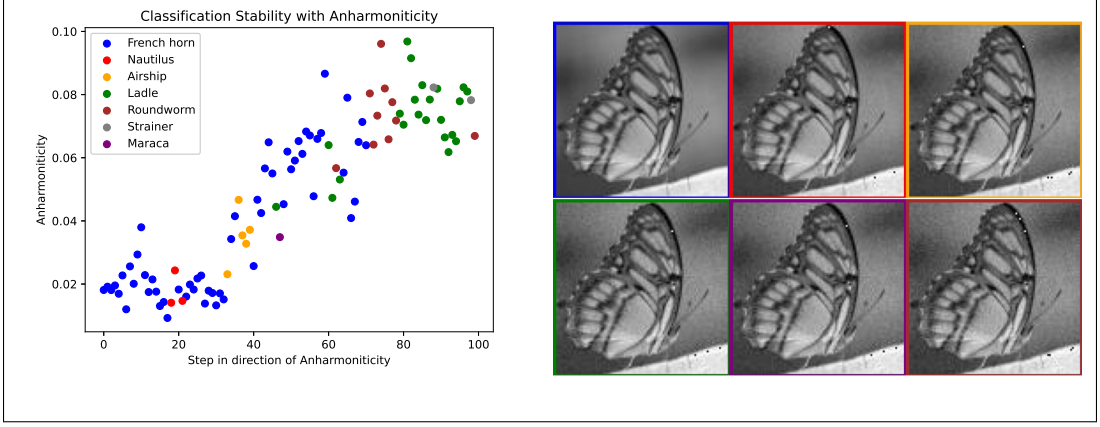

Figure 3: Demonstration of adversarial search procedure: following the stochastically increasing gradient of anharmoniticity brings out classification instability. The image at step N differs from the original image by N pixels.

Note this procedure is actually stochastic gradient ascent as our γ-computation is based on random sampling of the hypercube.

Algorithm 2 γ-Stochastic Adversarial Search at a point x in feature space

procedure ADVERSARIALSEARCH(x,r,N)
  currPoint $\leftarrow x$
  numSteps $\leftarrow N$
  for `each` step `in` numSteps **do**
    currGammas $\leftarrow \{\}$
    ballPoints $\leftarrow$ Ball(currPoint, r)
    for `each` point `in` ballPoints **do**
      currGammas[point] $\leftarrow \gamma$(point, r)
    end for
    currPoint $\leftarrow argmax$(currGammas)
  end for
  return currPoint
end procedure

We chose 25 as the number of steps to execute as preliminary experiments showed this is generally the number of pixels one needs to change for the data and models under review before adversarial examples appear. Figure 3 shows one of such experiments where we execute the adversarial search for 100 steps, the original class label changing ever more frequently along that path of (stochastically) increasing γ. This procedure is actually very effective for quickly and reliably finding adversarial attacks on any input image, e.g., see Figure 4 where we show examples from each class where the predicted class radically changes after changing just 25 scattered pixels according to our adversarial scheme. Presumably these models behave more stably on larger color images, but it is useful to see how they behave on out-of-domain data, and γ gives you a way to measure that.

The reader may consult the Appendix for results on the full 1000 images, which we summarize here as supporting the conclusion that the generally more accurate ViT is also more robust(stable) than ResNet-50, except for the Cow and Squirrel class, and that γ, or rather γ in combination with the predicted class probability itself, accurately predicts this pattern as follows: the class softmax probability $\mathcal{P}_C$, controlled by the class logit (L_C) and average logit ($\overline{L}$),

$$Prob(softmax)_C \equiv \mathcal{P}_C \equiv \frac{e^{L_C}}{\Sigma_{i=1..N_{classes}} e_i^L} \approx \frac{e^{L_C}}{e^{L_C} + (N_{classes} - 1) \cdot e^{\overline{L}}} \tag{4}$$

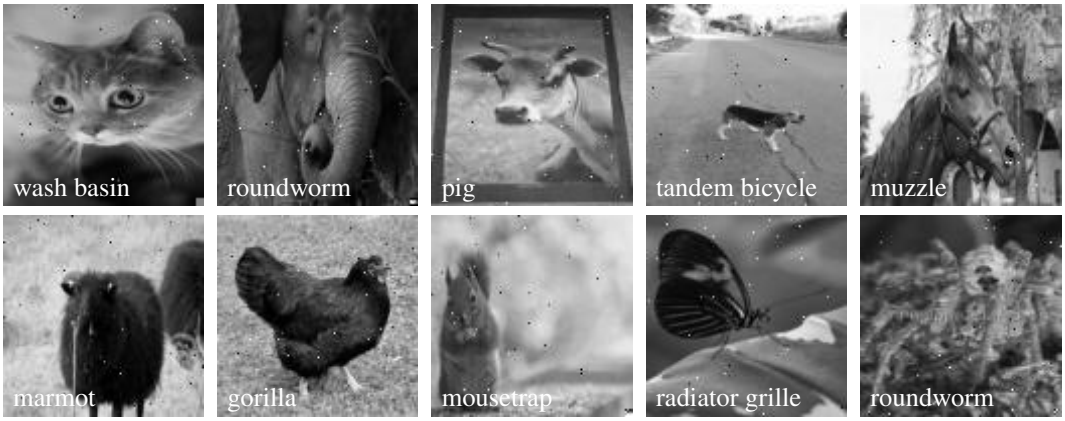

Figure 4: Examples of adversarial examples in ResNet-50 from following gradient of γ for 25 steps. Each image was originally correctly classified, but changed classes with modification of 25 scattered pixels as shown.

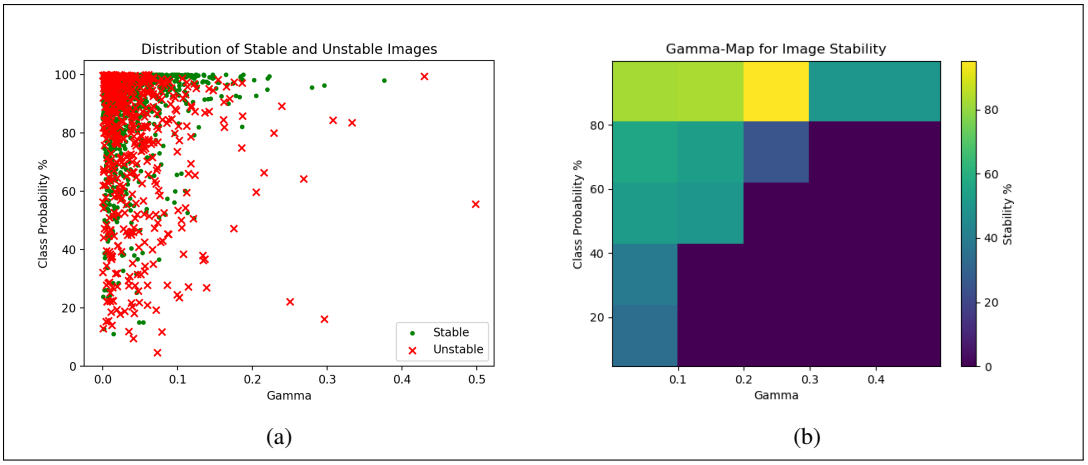

Figure 5: (a) Plotting predicted class probability $\mathcal{P}$ and γ for 1000+1000 images classified with ResNet and ViT shows the unstable images tend to dominate high-γ/low-$\mathcal{P}$ regions. (b) Density version of the previous plot serves as a practical "Gamma Map".

after a certain number N of gradient steps, is reduced by virtue of the class logit decreasing, on average, from L_C to $L_C - N\gamma$, so the adjusted probability becomes

$$\mathcal{P}'_C \approx \frac{e^{L_C - N\gamma}}{e^{L_C - N\gamma} + (N_{classes} - 1) \cdot e^{\overline{L}}} \approx \mathcal{P}_C e^{-N\gamma} \tag{5}$$

This metric, $\mathcal{P}_C e^{-N\gamma}$, gives a sort of “N-step adversarial robustness" which one can immediately measure at inference time and correlates well with actual image stability. We obtain visual confirmation of this metric by plotting the measured values of Class Probability $\mathcal{P}$ and γ for each image, as well as whether it is stable or not after N=25 iterations, obtaining a “Gamma Map" (Figure 5). As a practical tool, this type of plot allows one to immediately gauge whether a predicted classification is likely to be stable just from measuring $\mathcal{P}$ and γ, i.e., without having to do a full adversarial search.

6 Discussion and Conclusion

The foregoing demonstrated computation and interpretation of anharmoniticity (γ) as a robustness(stability) metric for both a low-dimensional training setting, where γ can act as a regulator and rank models by degree-of-overfitting, as well as a rather different high-dimensional inference

setting where γ can feed into real-time performance feedback — from this the reader can interpolate and extend to other scenarios.

Algorithmic implementation of γ is quite simple, straightforward, and applicable to any model function as a measure of robustness. Monitoring systems and testing procedures can easily integrate computations of γ as an alerting and regression test mechanism, respectively; for as we saw above for ViT and ResNet, accuracy does not imply robustness. We see no reason why model builders should hold back from computing γ alongside usual validation loss and other metrics to control overfitting. As a proxy for explainability, this may lead into incorporating other metrics for Responsible AI into the model life-cycle. As a metric to publish with a model's quality card, one can envision reporting γ for different data sets, indicating where a model is expected to more perform robustly. As a standard for ML model quality, functional standards are translatable, shareable, and optimizable across the industry; they may even point to certain mathematical truths pertaining to optimal ML systems.

We close with emphasizing a possibly trailblazing facet of our work: that one may apply a functional mathematical standard, i.e., conformity to the properties of harmonic functions, to a ML system as a way of assessing its quality and propriety for public usage. The harmonic standard may not be necessarily ideal, most real-world ML functions being far from harmonic, but we posit that it is better to reference a mathematically sound standard than having no such standard at all, giving AI systems free reign in their inner complexity while relying on conventional external metrics like precision and recall for quality control. For, assuming the white- or gray-box environment is not always going to be available to us, if we do not devise multiple ways to check models' inner complexity in a black-box environment, we will be giving up too much control over what these systems may surprise us with.

References

[1] Frederik Brasz. Soap Films: Statics and Dynamics (available at https://www.princeton.edu/~stonelab/Teaching/FredBraszFinalPaper.pdf). 2010.

[2] https://www.britannica.com/science/electricity/Deriving-electric-field-from-potential.

[3] https://www.mathworks.com/matlabcentral/fileexchange/85073-2d-transient-heat-conduction.

[4] Roel Bouman, Zaharah Bukhsh, and Tom Heskes. Unsupervised anomaly detection algorithms on real-world data: how many do we need? *Journal of Machine Learning Research*, 25(105):1–34, 2024.

[5] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre Alvise-Rebuffi, Ira Ktena, Krishnamurthy Dvijotham, and Taylan Cemgil. A fine-grained analysis on distribution shift. *arXiv preprint arXiv:2110.11328*, 2021.

[6] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

[7] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. *Advances in neural information processing systems*, 29, 2016.

[8] Jun Guo, Wei Bao, Jiakai Wang, Yuqing Ma, Xinghai Gao, Gang Xiao, Aishan Liu, Jian Dong, Xianglong Liu, and Wenjun Wu. A comprehensive evaluation framework for deep model robustness. *Pattern Recognition*, 137:109308, 2023.

[9] Joana C Costa, Tiago Roxo, Hugo Proença, and Pedro RM Inácio. How deep learning sees the world: A survey on adversarial attacks & defenses. *arXiv preprint arXiv:2305.10862*, 2023.

[10] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] Beilun Wang, Ji Gao, and Yanjun Qi. A theoretical framework for robustness of (deep) classifiers against adversarial examples. *arXiv preprint arXiv:1612.00334*, 2016.

[13] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, Ian Goodfellow, and G Brain. The relationship between high-dimensional geometry and adversarial examples. *arXiv preprint arXiv:1801.02774*, 2018.

[14] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *Advances in neural information processing systems*, 31, 2018.

[15] Micah Goldblum, Avi Schwarzschild, Ankit Patel, and Tom Goldstein. Adversarial attacks on machine learning systems for high-frequency trading. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

[16] Constantine Caramanis, Shie Mannor, and Huan Xu. Robust optimization in machine learning. 2011.

[17] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *Advances in neural information processing systems*, 29, 2016.

[18] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Robustness of classifiers to universal perturbations: A geometric perspective. *arXiv preprint arXiv:1705.09554*, 2017.

[19] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 278–287, 2017.

[20] Giorgio Talenti. A note on the gauss curvature of harmonic and minimal surfaces. *Pacific Journal of Mathematics*, 101(2):477–492, 1982.

[21] Forina Mea. PARVUS—an extendible package for data exploration, Classification and Correlation Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy. Data available at https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html, 1991.

[22] https://huggingface.co/microsoft/resnet-50.

[23] https://huggingface.co/docs/transformers/model_doc/vit.

[24] https://www.kaggle.com/datasets/alessiocorrado99/animals10.

[25] Yuhang Wu, Sunpreet S Arora, Yanhong Wu, and Hao Yang. Beating attackers at their own games: Adversarial example detection using adversarial gradient directions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2969–2977, 2021.