Chaos Theory and Adversarial Robustness

Anonymous authors
Paper under double-blind review

Abstract

Neural Networks, being susceptible to adversarial attacks, should face a strict level of scrutiny before being deployed in critical or adversarial applications. This paper uses ideas from Chaos Theory to explain, analyze, and quantify the degree to which Neural Networks are susceptible to or robust against adversarial attacks.

Our results show that susceptibility to attack grows significantly with the depth of the model, which has significant safety implications for the design of Neural Networks for production environments. We also demonstrate how to quickly and easily approximate the certified robustness radii for extremely large models, which until now has been computationally infeasible to calculate directly, as well as show a clear relationship between our new susceptibility metric and post-attack accuracy.

1 Introduction

The current state of Machine Learning research presents Neural Networks as black boxes due to the high dimensionality of their parameter space, which means that understanding what is happening inside of a model regarding domain expertise is highly nontrivial, when it is even possible. However, the actual mechanics by which Neural Networks operate - the composition of multiple nonlinear transforms, with parameters optimized by a gradient method - were human-designed, and as such are well understood. In this paper, we will apply this understanding, via analogy to Chaos Theory, to the problem of explaining and measuring susceptibility of Neural Networks to adversarial methods.

It is well-known that Neural Networks can be adversarially attacked, producing obviously incorrect outputs as a result of making extremely small perturbations to the input (Goodfellow et al., 2014; Szegedy et al., 2013). Prior work, like Shao et al. (2021); Wang et al. (2018) and Carmon et al. (2019) discuss “Adversarial Robustness" in terms of metrics like accuracy after being attacked or the success rates of attacks, which can limit the discussion entirely to models with hard decision boundaries like classifiers, ignoring tasks like segmentation or generative modeling (He et al., 2018). Other work, like Li et al. (2020) and Weber et al. (2020), develop “certification radii," which can be used to guarantee that a given input cannot be misclassified by a model without an adversarial perturbation with a size exceeding that radius. However, calculating these radii is computationally onerous when it is even possible, and is again limited only to models with hard decision boundaries.

Regarding the existence of adversarial attacks in the first place, Pedraza et al. (2020) and Prabhu et al. (2018) have explained this behaviour of Neural Networks on the basis that they are dynamical systems, and then use some results from that analysis to try and classify adversarial inputs based on their Lyapunov exponents. However, this classification methodology rests on shaky ground, as the Lyapunov exponents of a single input must be relative to those of similar inputs, and it is entirely reasonable to imagine a scenario in which an input does not become more susceptible to attack just because it is itself adversarial.

In this work, we re-do these Chaos Theoretic analyses in order to understand, not particular inputs, but the Neural Networks themselves. We show that Neural Networks are dynamical systems, and then continuing that analogy past where Pedraza et al. (2020) and Prabhu et al. (2018) left off, investigate what Neural-Networks-as-dynamical-systems means for their susceptibility to attack, through a combination of analysis and experimentation. We develop this into a theory of Adversarial Susceptibility, the “Susceptibility Ratio"

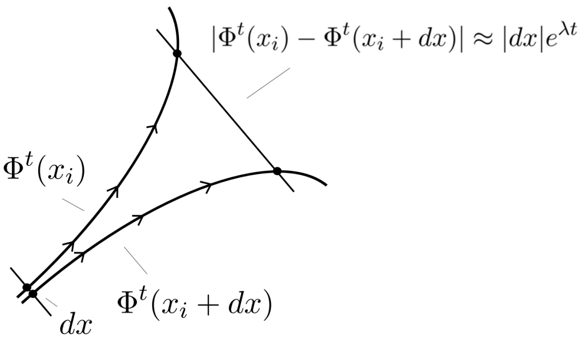
$|\Phi^t(x_i) - \Phi^t(x_i + dx)| \approx |dx|e^{\lambda t}$
$\Phi^t(x_i)$
$\Phi^t(x_i + dx)$
dx

Figure 1: In a dynamical system, two trajectories with similar starting points may, over time, drift farther and farther away from one another, typically modeled as exponential growth in the distance between them. This growth characterizes a system as exhibiting “sensitive dependence," known colloquially as the “butterfly effect," where small changes in initial conditions eventually grow into very large changes in the eventual results.

as a measure of how effective attacks will be against a Neural Network, and show how to easily numerically approximate this value. Returning to the work in Li et al. (2020) and Weber et al. (2020), we use the susceptibility ratio to quickly and accurately estimate the certification radii of very large Neural Networks, aligning this paper with prior work.

2 Neural Networks as Dynamical Systems

We’ll start by re-writing the conventional feed-forward Neural Network in the language of dynamical systems, and then transferring the analysis of dynamical systems over to Neural Networks. But we should begin by explaining what a dynamical system is (Alligood et al., 1998).

2.1 Dynamical Systems

In Chaos Theory, a dynamical system is composed of three core ingredients. Ingredient one is T, representing “time," or something like it. A change in time can be added to an initial time to get an end time, in an associative fashion. Ingredient two is X, the state space. Elements of X could include the positions of a pendulum, the states of memory in a computer program, or all the possible arrangements of atoms in a room. And the third ingredient is $\Phi : T \times X \to X$, the “evolution function" of the system. When Φ is given a state $x_{i,t}$ and a change in time Δt, it returns $x_{i,t+\Delta t}$, which is the new state of the system after Δt time has elapsed. We’ll write this as

$$x_{i,t+\Delta t} = \Phi(\Delta t, x_{i,t})$$

In order to stay well defined, this has to have some properties like being consistent with the end result regardless of how intermediate states were taken, like so:

$$\Phi\big(\Delta t_a, \Phi(\Delta t_b, x_{i,t})\big) = \Phi(\Delta t_a + \Delta t_b, x_{i,t})$$

From this, we can take a “trajectory" of the initial state $x_{i,0}$ over time, with points represented by $\big(t, \Phi(t, x_{i,0})\big)$. In order to simplify the notation, and following on from the notion that the evolution over time of a system can be thought of as the composition of multiple instances of the evolution function, we will write this trajectory as

$$\Phi(t, x_{i,0}) = \Phi^t(x_i)$$

The final piece of the puzzle, the Chaos in Chaos Theory, concerns the relationship between trajectories with very similar initial conditions, say x_i, and $x_i + dx$, where dx is some very small change, such as subtly reorienting the arms of a double pendulum before setting it into motion. We then need some notion of the distance between two elements of the state space, but we will assume that the space is some sort of vector space equipped with a notion of length written with $|\cdot|$, and proceed from there. For the initial condition, we know off the bat that

$$|\Phi^0(x_i) - \Phi^0(x_i + dx)| = |dx|$$

However, the interesting analysis comes when we model this difference as time progresses. In some systems, very small differences in the initial condition end up being ignored, such as the position of an oscillator with a damping force; no matter what, you reach the resting state, and that's the end of the story. However, in some systems, very small differences in the initial condition end up compounding on themselves, like the flaps of a butterfly's wings eventually resulting in a hurricane. Both of these can be approximately modeled by an exponential function, like so

$$|\Phi^t(x_i) - \Phi^t(x_i + dx)| \approx |dx|e^{\lambda t}$$

In each of these cases, the growing or shrinking differences between the trajectories are described by λ, also called the Lyapunov exponent. If $\lambda < 0$, these differences disappear over time, and the trajectories of two similar initial conditions will eventually align with one another. However, if $\lambda > 0$, these differences increase over time, and the trajectories of two similar initial conditions will grow farther and farther apart, until they might as well have started from entirely different regions of the state space. This is called "sensitive dependence," and is the mark of a chaotic system. It must be noted, however, that the exponential nature of this growth is a shorthand model, with obvious limits, and is not fully descriptive of the underlying behavior.

2.2 Neural Networks

Conventionally, a Neural Network is given a formulation along the following lines (Schmidhuber, 2015). It is given by a function $h : \Theta \times X \to Y$, where Θ is the space of possible learned parameters with W_l being multiplicative weight matrices and b_l as additive bias vectors, X is the vector space of possible inputs, and Y is the vector space of possible outputs. Each of the L layers in the Neural Network is given by a matrix multiplication, an optional bias addition, and a nonlinear activation function, with hidden states z_l representing the intermediate values taken during the inference operation, e.g.

$$z_{i,0} \coloneqq x_i$$

$$z_{i,l+1} = \sigma(W_l z_{i,l} + b_l) | W_l, b_l \subset \theta$$

$$h(\theta; x_i) = \hat{y}_i \coloneqq z_{i,L}$$

Now, consider rewriting this by saying that a Neural Network is a dynamical system composed of three ingredients. Ingredient one is $[L] = \{0, 1, 2 \ldots L\}$, which here will be used to represent the current depth of the hidden state, from 0 for the initial condition up to L for the eventual output. Ingredient two is Z, which is the vector space of all possible hidden states. And ingredient three is $g : [L] \times Z \to Z$, which is written here as

$$z_{i,l+1} = g(1, z_{i,l}) = \sigma(W_l z_{i,l} + b_l)$$

We will handwave the method by which g "knows" which parameters W_l and b_l to use, perhaps using something along the lines of a dimension appended to $z_{i,l}$ that records the current value of l, and which

g iterates, rather than including in the ordinary operations of the feed-forward layer. The generalization to $g(\Delta l, z_{i,l})$ then follows from the same rule of composition applied to the dynamical systems, at least for integer values of Δl, under the condition that it never leaves $[L]$. We can also then re-write the notation along the lines of that for the dynamical systems, e.g.

$$g(l, z_{i,0}) = g^l(x_i)$$

Noting of course that we have defined $z_{i,0}$ as x_i. From here, we can start to discuss the trajectories of the hidden states of the Neural Network, and what happens when their inputs are changed slightly. For the first hidden state, defined as the input, we can immediately say that

$$|g^0(x_i) - g^0(x_i + dx)| = |dx|$$

And then by once again mapping to the dynamical systems perspective, we model the difference between the two trajectories at depth l with

$$|g^l(x_i) - g^l(x_i + dx)| \approx |dx|e^{\lambda l}$$

When the value of λ is greater than 0, we call the Neural Network sensitive to the input, but when the value of $e^{\lambda L}$ - essentially, the ratio of the size of the change of the output to the size of the change in the input - is very large, we call dx an Adversarial Perturbation. If this analogy holds, we should expect that when we adversarially attack a Neural Network, the difference between the two corresponding hidden states should grow as they progress through the model. Again, as per the dynamical system, this growth is not necessarily exponential, but using an exponential model is the most illustrative. This is our first experimental result.

3 Experimental Design

For our experiments, we used two different model architectures: ResNets (He et al., 2015), as per the default Torchvision implementation (Marcel & Rodriguez, 2010), and a custom CNN architecture in order to have finer-grained control over the depth and number of channels in the model. The ResNets were modified, and the custom models built as to allow for recording all of the hidden states during the inference operation. These models, unless specified that they were left untrained, were trained on the Food-101 dataset (Bossard et al., 2014) for 50 epochs with a batch size of 64 and a learning rate of 0.0001 with the Adam optimizer against Cross Entropy Loss. The ResNet models used were ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152.

In the Torchvision ResNet class, models consist of operations named *conv1, bn1, relu, maxpool, layer1, layer2, layer3, layer4, avgpool,* and and *fc*, with the first four representing a downsampling intake, then four more “blocks" of ResNet layers, and then a final operation that converts the 3D spatial tensor into a 1D class weight tensor. Hidden states are recorded at the input, after *conv1, layer1, layer2, layer3, layer4,* and the output.

The custom models, specified with C and D, consist of D tuples of convolutional layers, batch normalization operations, and ReLU nonlinearities, with the first tuple having a downsampling convolution and a maxpool operation after the ReLU. Each of these convolutions, besides the first which takes in three channels, has C hidden layers. Finally, there is a 1×1 convolution, a channel-wise averaging, and then a single fully connected layer with 101 outputs, one for each class in the Food-101 dataset. Hidden states are recorded after every tuple, and also include the input and the output of the model. The first tuple approximates the downsampling intake of the ResNet models.

In order to better handle the high dimensionality and changes in scale of the inputs, outputs, and hidden states, rather than using the Euclidean $L2$ norm as the distance metric, we used a modified Euclidean distance

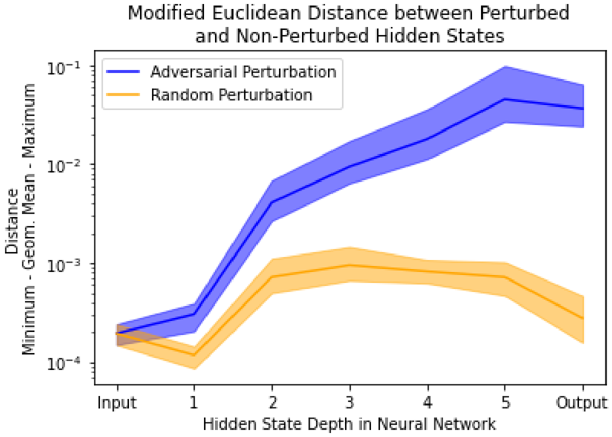

Figure 2: Example of hidden state drift while performing inference with the ResNet18 model. Note the logarithmic scaling on the y-axis.

$$|\vec{v}| := \sqrt{\frac{1}{\mathbf{dim}(\vec{v})} \sum_i v_i^2}$$

This will be applied to every instance of length, distance, radius, and so on. Adversarial perturbations dx_{adv} against a Neural Network $h(\theta; \cdot)$ of a given radius r for a given input x_i were generated by using five steps of gradient descent with an update size of 0.01, maximizing

$$|h(\theta; x_i) - h(\theta; x_i + dx_{adv})|$$

and projecting back to the hypersphere of radius r after every update. These attacks are along the lines of Zhang et al. (2021); Wu et al. (2021; 2022); Xie et al. (2021) and Shao et al. (2021), and their use of attacks with l_p-norm decay metrics or boundaries. For comparison, random perturbations were also generated, by projecting randomly sampled Gaussian noise to the same hypersphere. In order to perform these experiments under optimal conditions, the inputs that were adversarially perturbed were selected only from the subset of the Food-101 testing set for which every single model trained agreed on the top-1 output class and were correct. A Jupyter Notebook implementing these training regimes and attacks will be made available alongside this manuscript, pending review.

4 Hidden State Drift

An example of the approximately exponential growth in the distance between hidden states between normal and adversarially perturbed inputs hypothesized in section 2.2 for 32 inputs is shown in Figure 2. Between the initial perturbations, generated with a radius of 0.0001, and the outputs, the differences grew by a factor of $\sim 747\times$. Given that ResNet18 has 18 layers, using $747 \approx e^{18\lambda}$, we can calculate $\lambda \approx 0.368$, a measure of this drift per layer. However, the Lyapunov exponent for each layer is of less interest to an adversarial attacker or defender, with the actual value of interest being given by this new metric, ψ, the adversarial susceptibility for a particular input and attack, given by

$$\psi(h, \theta, x_i, dx_{adv}) := e^{\lambda L} = \frac{|h(\theta; x_i) - h(\theta; x_i + dx_{adv})|}{|dx_{adv}|} \tag{1}$$

Essentially, this measures the “Susceptibility Ratio," the ratio of the damage done by a given adversarial perturbation to its original size. If this is a meaningful metric by which to judge a Neural Network architecture, it should remain relatively stable despite changes in the radius of the adversarial attack. This is

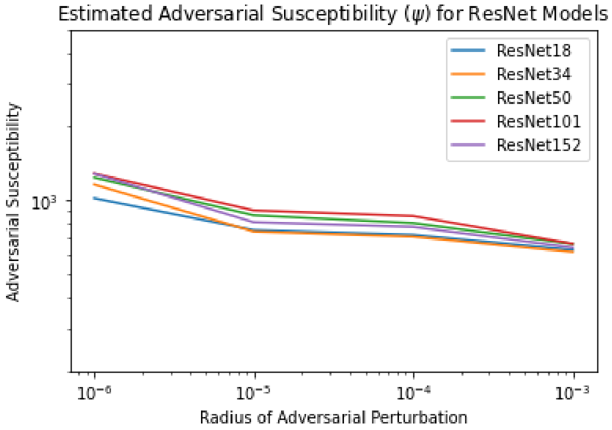

Figure 3: Despite a change in the radius of the adversarial perturbation by three orders of magnitude, the value of ψ associated with those attacks remains relatively stable.

	ResNet18	ResNet34	ResNet50	ResNet101	ResNet152
$\hat{\Psi}(h,\theta)$	781.2	790.7	854.4	893.2	846.5

Table 1: Overall Adversarial Susceptibility of trained ResNet models.

our second experimental result, demonstrated in Figure 3. By sampling ψ over a number of inputs x_i and a variety of attack radii $|dx_{adv}|$ and taking the geometric mean[1], we can come to a single value, written as

$$\Psi(h,\theta) = e^{\mathbb{E}[\ln(\psi(h,\theta,x_i,dx_{adv}))]}$$

and approximated with $\hat{\Psi}(h,\theta)$, giving a measure of the adversarial susceptibility for the model as a whole. These values have been calculated for the trained ResNet models, and are given in Table 1. These results contradict the predictions that we will make in the next section, at which point we will begin using the custom model architectures to begin to tease out the relationships between a Neural Network's architecture and its Adversarial Susceptibility.

$\hat{\Psi}(h,\theta)$

		Channels (C)			
		32	64	128	256
Layers (D)	2	0.749	0.523	0.651	0.560
	4	1.021	0.695	0.775	0.610
	8	2.788	2.134	1.505	1.276
	16	15.491	12.935	8.423	7.123
	32	109.340	135.472	98.834	92.404
	64	96.037	63.785	60.443	48.721

Table 2: Adversarial Susceptibility of randomly initialized convolutional models with custom architectures on inputs consisting of random noise

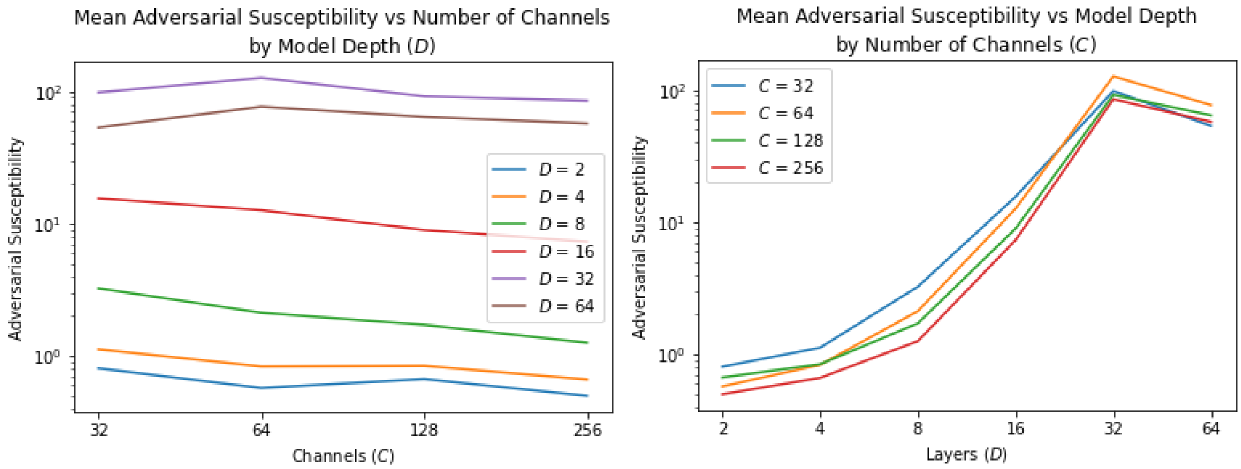

Figure 4: Graphical replication of Table 2

5 Architectural Effects on Adversarial Susceptibility

Returning to the definition of ψ given in equation 1, we might model it as being proportional to the exponent of L, the depth of the Neural Network. And yet, despite ResNet152 having more than eight times as many layers as ResNet18, its Adversarial Susceptibility is only marginally higher. Thus, use of an exponential model, at least to explain these experimental results, is limited. In order to explore this reasoning further in a more numerically ideal setting, we present our third experimental result, in Table 2, and replicated in Figure 4. Here, using randomly initialized, untrained models with custom architectures as described in the experimental methods section, giving them random inputs, and then attacking them on those inputs, we can tease apart the relationship between model architecture and Adversarial Susceptibility, in the case where both parameters and input dimensions are normally distributed.

We immediately find an approximately exponential relationship between the Adversarial Susceptibility and the depth of the model that was expected based on equation 1, however the slight dip upon moving from 32 to 64 layers is unexpected, and while exploring its potential causes and implications is outside of the scope of this paper, it may warrant further experimentation and analysis.

Also of interest is the effect, or lack thereof, of increasing the number of channels in the Neural Network. While a quadratic increase in the number of parameters in the model might be expected to increase its Adversarial Susceptibility, especially in the absence of batch normalization operations, no experiment that we performed yielded such a result. Our initial hypothesis followed this line of reasoning; taking two matrices A and B composed of standard Gaussian noise and setting $C = AB$, the standard deviation over all of the entries in C is proportional to the square root of the dimension shared by A and B, Analogizing A to the weights of a layer, B to a small change in the hidden state, and C to the resulting change in the next hidden state, we expected the Lyapunov exponent produced by a model to be proportional to the logarithm of the square root of the number of channels of that layer. But no such experimental evidence could be found for this.

We repeated the Adversarial Susceptibility testing on the same model architectures, this time with trained parameters, and with inputs from Food-101 that every single model agreed and was correct on. These results are in Table 3, and replicated in Figure 5.

The largest difference here is that, for every model, the susceptibility has increased by an order of magnitude, if not several. Training and switching to a domain that contains information relevant to the model has resulted in it being far, far more sensitive to attack. Yet, following up on earlier experiments, we can

---

[1] In order to increase the numerical stability of the geometric mean calculation, we use $\sqrt[n]{\prod_{i=0}^{n} a_i} = e^{\frac{\sum_{i=0}^{n} \ln a_i}{n}}$

$\hat{\Psi}(h, \theta)$

		Channels (C)			
		32	64	128	256
Layers (D)	2	578.602	610.207	586.503	576.759
	4	1470.399	1658.209	1631.561	1695.993
	8	2144.418	2224.467	2536.745	2370.648
	16	2361.251	2401.381	2485.030	2846.418
	32	3162.568	3018.758	2987.640	3256.967
	64	2045.765	2213.575	3103.335	2471.823

Table 3: Adversarial Susceptibility of trained convolutional models with custom architectures on inputs consisting of Food-101 samples that every model agreed on and were correct.

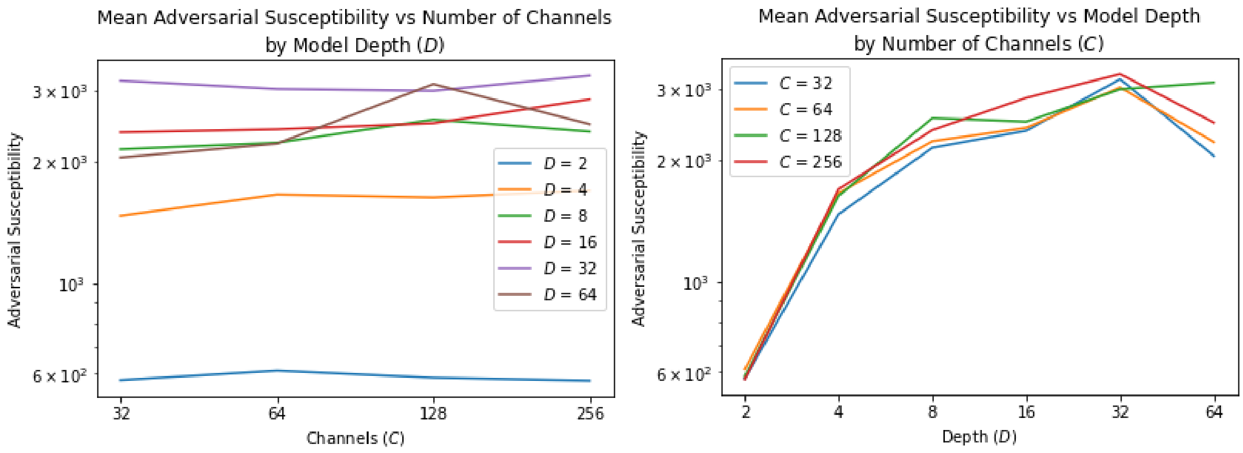

Figure 5: Graphical replication of Table 3

again see that the number of channels doesn't effect the Adversarial Susceptibility of the model much, while the number of layers increases it significantly. However, this time, the relationship between the number of layers and the susceptibility has changed, to be almost logarithmic, rather than approximately exponential in nature, and somewhat replicates the relationship found between depth and susceptibility in the trained ResNet models. Interestingly, this is reasonably analogous to the testing accuracy of the models, where increases in depth yield diminishing returns, and it may be theorized that both of these effects are due to changes in the distributions of the weights. However, it must be noted that the increase in susceptibility is greater than the increase in accuracy. Making models deeper makes them more vulnerable faster than it makes them better, with additional costs in memory, runtime, and energy consumption.

6 Relationships to Other Metrics

6.1 Approximation of Certified Robustness Radii

In the work of Weber et al. (2020) and Li et al. (2020), they attempt to calculate what they refer to as "Certified Robustness Radii." For a model with hard decision boundaries, e.g. a top-1 classification model, its Certified Robustness Radius is the largest value ϵ_h such that, for any input x_i and any adversarial perturbation $|dx_{adv}|$, the ultimate classification given by the model $\text{argmax}_c h(\theta; x_i) = \text{argmax}_c h(\theta; x_i + dx_{adv})$ for all perturbations with radius smaller than ϵ_h. In their work, however, they state explicitly that these values are incredibly computationally demanding to calculate for small models, and computationally infeasible for larger models. However, using the Adversarial Susceptibility for a model, one can quickly approximate this certified robustness radius for even very large models. It is simply the distance to the nearest decision boundary, divided by the Adversarial Susceptibility.

Consider the following example: a five-class model outputs the following weights for a given input, $\hat{y} = \{2.1, 0.6, 0.1, -0.5, -1.1\}$. Thus, the nearest decision boundary occurs where the first and second classes become equal, at $\hat{y}' = \{1.35, 1.35, 0.1, -0.5, -1.1\}$. The modified Euclidean distance between these two is 0.4703. Suppose that this model has an Adversarial Susceptibility of $\hat{\Psi} = 25.0$. Its certified robustness radius would then be estimated as $\epsilon = \frac{0.4703}{25.0} = 0.01897$. One could then take the mean or minimum over these values for every input in a dataset, and a number could be produced for the model as a whole.

Finally, an over-all criticism has to be made regarding the use of these certified robustness radii in general. Consider two models used for a binary classification problem, inferring on the same input, which has been perturbed by adversarial attacks of equal radii. The first model, moving from the vanilla to the adversarial input, changes its output from $\{0.9, 0.1\}$ to $\{0.6, 0.4\}$. The second model, under the same conditions, changes its output from $\{0.55, 0.45\}$ to $\{0.45, 0.55\}$. Using a certified robustness radius, you would say that the first model is the more robust, while a more direct reading of the change in probabilities would declare the second model to be more robust. These certified robustness radii wrap a lot of information about the model together with information about the inputs and the distributions they are drawn from together, so it can be difficult to use them as a metric. Imagine if, in the previous example, the first model was only so confident because it was massively overfit, and the actual input is relatively non-separable. Although this improves its robustness radius, it makes it more susceptible to attack in the field.

6.2 Post-Adversarial Accuracy

One of the more standard measures of adversarial robustness is to measure the accuracy of models on adversarially perturbed inputs. If our analysis and experimental results thus far are correct, we should see an inverse relationship between measured Adversarial Susceptibility and the post-adversarial accuracy for any given attack radius. This is our fourth experimental result, shown in Figure 6. In it, we see that, among ResNets, which all had very similar values of $\hat{\Psi}(h, \theta)$, post-attack accuracies are relatively similar between models, with an approximate but minor correspondence between higher susceptibilities and lower post-attack accuracy. We also see, among the custom architectures, represented in Figure 6 by the subset of models with 32 channels and in their entirety in Figure 4, a very close inverse relationship between higher susceptibility and lower post-attack accuracy, especially at the 0.01 attack radius. We also see that the

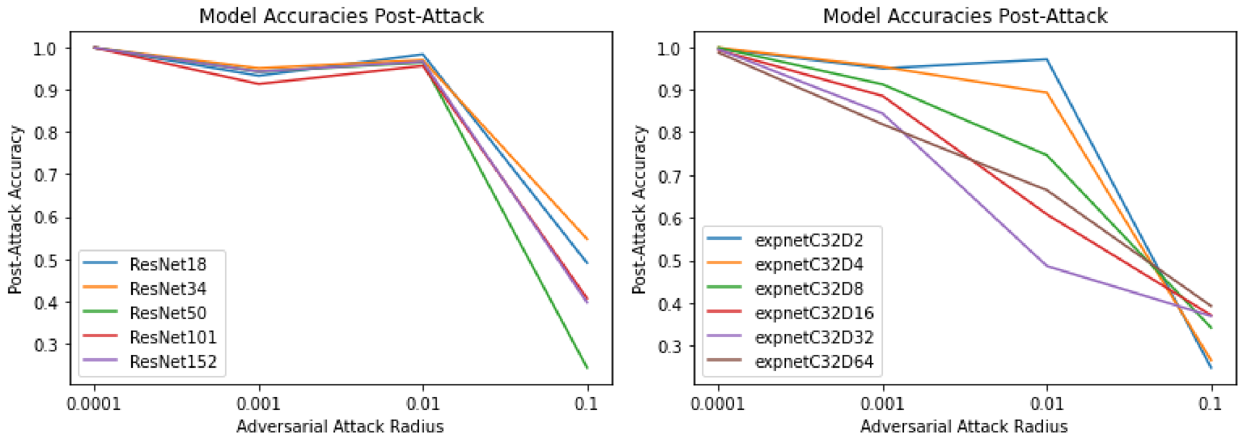

Figure 6: Post-Attack Accuracies

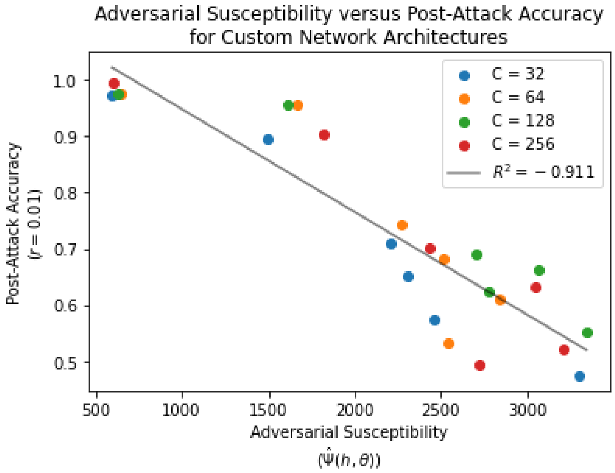

Figure 7: Relationship between Adversarial Susceptibility and Post-Attack Accuracy, with a radius of 0.01. Linear best fit shown, with a correlation coefficient of -0.911.

custom architecture with $D = 2$, which experimentally had $\hat{\Psi} = 578.602$, has a post-attack accuracy curve that very closely resembles those of the ResNet models, each of which had a similar susceptibility.

7 Conclusions and Future Work

Our experiments have shown, with some variation due to the inscrutable black-box nature of Deep Learning, that there is an extremely strong, analytically valuable, and experimentally valid connection between Neural Networks and dynamical systems as they exist in Chaos Theory. We can use this connection to make accurate and meaningful predictions about different Neural Network architectures, as well as efficiently measure how susceptible they are to adversarial attacks. We have shown a correspondence, both experimentally and analytically, between these new measurements, and those developed in prior works. Thus, a new tool has been added to the toolbox of practitioners looking to make decisions about Neural Networks.

Future work will include further exploration in this area, and the pulling in of more advanced techniques and analysis from Chaos Theory, as well as the development of new, more precise metrics that tell us even more about how models are effected by adversarial attacks. Additionally, the relationship between Adversarial

Susceptibility and adversarial robustness training regimes deserves study, as well as the relationship with different attack methodologies.

References

Kathleen T Alligood, Tim D Sauer, James A Yorke, and David Chillingworth. Chaos: an introduction to dynamical systems. *SIAM Review*, 40(3):732–732, 1998.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018.

Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. *arXiv preprint arXiv:2009.04131*, 2020.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pp. 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL `https://doi.org/10.1145/1873951.1874254`.

Anibal Pedraza, Oscar Deniz, and Gloria Bueno. Approaching adversarial example classification with chaos theory. *Entropy*, 22(11), 2020. ISSN 1099-4300. doi: 10.3390/e22111201. URL `https://www.mdpi.com/1099-4300/22/11/1201`.

Vinay Uday Prabhu, Nishant Desai, and John Whaley. On lyapunov exponents and adversarial perturbation, 2018. URL `https://arxiv.org/abs/1802.06927`.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of vision transformers. *arXiv preprint arXiv:2103.15670*, 2021.

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.

Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5133–5142. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/wang18c.html`.

Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

Fan Wu, Linyi Li, Zijian Huang, Yevgeniy Vorobeychik, Ding Zhao, and Bo Li. Crop: Certifying robust policies for reinforcement learning through functional smoothing. *arXiv preprint arXiv:2106.09292*, 2021.

Fan Wu, Linyi Li, Chejian Xu, Huan Zhang, Bhavya Kailkhura, Krishnaram Kenthapadi, Ding Zhao, and Bo Li. Copa: Certifying robust policies for offline reinforcement learning against poisoning attacks. *arXiv preprint arXiv:2203.08398*, 2022.

Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. Crfl: Certifiably robust federated learning against backdoor attacks. In *International Conference on Machine Learning*, pp. 11372–11382. PMLR, 2021.

Chaoning Zhang, Philipp Benz, Chenguo Lin, Adil Karjauv, Jing Wu, and In So Kweon. A survey on universal adversarial attack. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2021. doi: 10.24963/ijcai.2021/635. URL `https://doi.org/10.24963%2Fijcai.2021%2F635`.