Towards Unifying Smooth Neural Codes with Adversarially Robust Representations

Rylan Schaeffer
Institute for Applied Computational Science
Harvard University
Cambridge, MA 02138
rylanschaeffer@g.harvard.edu

Helena Casademunt
Department of Physics
Harvard University
Cambridge, MA 02138
hcasademunt@g.harvard.edu

Nayantara Mudur
Department of Physics
Harvard University
Cambridge, MA 02138
nmudur@g.harvard.edu

Abstract

Recent work has linked the geometry of neural population codes to their eigenspectra [31], yielding a hypothesis that the vulnerability of deep convolutional neural networks (CNNs) to adversarial attacks might be an emergent consequence of the networks learning fractal representations. We demonstrate experimentally on a diverse set of CNNs trained on ImageNet [7] that this hypothesis is false. We further demonstrate that artificial network codes are largely non-responsive to input images that deviate from the training data distribution, another key departure from the theory. Finally, we propose two loss regularization terms, implement both, and demonstrate that while one regularizes networks trained on CIFAR-10 [20] as expected, the other has no apparent effect. The regularization results raise questions concerning the relevance of this fractal bound.

1 Introduction

Recent work in computational neuroscience has sought to understand visual sensory processing by comparing convolutional neural networks (CNNs) to biological circuits at all levels of the visual hierarchy, from retina [25, 35] to V1/V2/V4 [12] to inferotemporal (IT) cortex [3, 17] to the entire visual ventral stream [22]. However, as one paper aptly notes, the main contribution of this line of work is demonstrating that artificial networks can explain a high fraction of variance in their biological counterparts, suggesting that absent scientific understanding, "we may simply be replacing one inscrutable blackbox (the brain), with another (a potentially overparameterized deep network)" [35].

To further an understanding of visual sensory processing, we sought to test a recent prediction concerning the geometry of neural population codes in artificial neural networks. Specifically, Stringer, Pachitariu et al. [31] argued that in order for a population of neurons to have a high dimensional but smooth representation of stimuli, the eigenspectrum of the covariance of the population must obey a power law such that $\lambda_n = O(n^{-\alpha^*})$, where λ_n is the n-th eigenvalue, $\alpha^* = 1 + 2/d$ and d is the dimension of the stimulus. For exponents α above this bound α^*, the neural response becomes fractal, meaning nearly identical stimuli in stimulus-space can be represented arbitrarily far apart in neural representation-space and vice versa.

Preprint. Under review.

Stringer, Pachitariu et al. give an example of a one-dimensional ($d = 1$) stimulus (e.g. drifting gratings), with an optimal power law given by $\alpha* = 1 + 2/d = 3$. If the eigenspectrum decays too slowly ($\alpha = 2$) (Fig 1, c, middle row), the representation necessarily becomes fractal in order to encode increasingly fine-grained information, visualized by the tangled random projection (Fig 1, c, bottom row). If the eigenspectrum decays too quickly ($\alpha = 4$) (Fig 1, e, middle row), the representation remains smooth, but information that could otherwise be preserved is not (Fig 1, c, bottom row). In the Goldilocks zone, the representation encodes as much information as possible while remaining smooth (Fig 1, c, middle and bottom row).

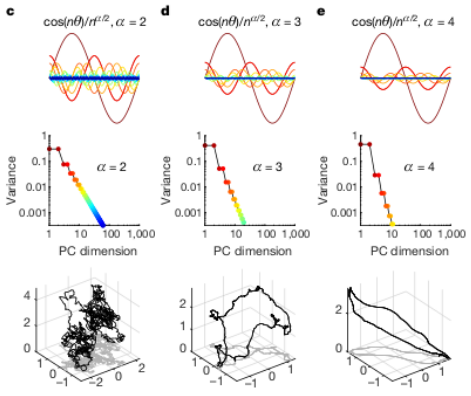

Figure 1: Adapted from Stringer, Pachitariu et al. Figure 4 [31]. Stimuli (top row) have coarse (red) to fine (blue) grained details. Encoding all details forces the neural code to become fractal (c, bottom). Conversely, a smooth representation (e, bottom) may lose stimulus information (e, middle). The optimal representation is a code as high dimensional as possible (d, middle) without trespassing into fractality (d, bottom).

The authors hypothesized this bound might explain why making small but intentional perturbations to input images can cause deep learning models to produce incorrect outputs with high confidence, a phenomenon termed adversarial attacks [10, 33]. The machine learning research community has devoted attention to defining, explaining and developing defenses against these adversarial examples; for a comprehensive review, see [37]. While other works have contributed to exploring and characterizing the geometry of adversarial examples [1, 23, 4, 9, 18, 5, 6], to the best of our knowledge, we are the first to test the hypotheses that emerge from Stringer, Pachitariu et al. [31]. Specifically, we sought to answer the following questions:

1. What are the eigenspectra of neural activity in CNNs trained on natural images? Per Stringer, Pachitariu, et al., do the CNN eigenspectra indicate that the learned representations are fractal?
2. How do eigenspectra of neural activity in CNNs change for input data that differ from the training data distribution? Specifically, how do (a) noise, (b) reduced dimension image, and (c) robust and non-robust images [24] affect the eigenspectra?
3. What modifications to loss functions are sufficient to create a Stringer-Pachitariu power-law decay? Does possessing this decay improve a network's robustness against adversarial attacks?

2 Methods

2.1 Datasets

In our experiments, we used two datasets: ImageNet [7] and CIFAR-10 [20]. ImageNet is a large image database of approximately 14 million natural images with over 20,000 labeled classes. Images are typically 3 channels by 256 pixels by 256 pixels. CIFAR-10 is a smaller image database of 60000 natural images with 10 labeled classes. Images are each 3 channels by 32 pixels by 32 pixels.

We also test models with four CIFAR10 datasets from Ilyas et al. (2019) [16]. Ilyas et al. argued that features can be disentangled into robust and non-robust features. A robust feature (or a γ-robustly useful feature) is a feature that remains correlated (or anti-correlated) above a threshold (γ), with the true label under a set of adversarial perturbations. A non-robust useful feature is one that is correlated with the true label but does not remain correlated under a perturbation, for any $\gamma > 0$. The authors further demonstrate that a dataset composed of only robust features possesses high accuracy on the original training dataset as well as high 'robust accuracy'– accuracy against adversarial attacks. Conversely, a dataset composed of only non-robust features exhibited high accuracy on the original training dataset but poor robust accuracy. The four datasets used here (Sections 3.2.3, 3.3) consist of

the two datasets composed of robust and non-robust features respectively, and two datasets consisting of adversarial examples toward a deterministic and a random class.

2.2 Models

When discussing ImageNet, we used the following CNNs: AlexNet [19], MnasNet [34], VGG [29], ResNet [13], GoogLeNet [32]. These models achieved state-of-the-art results on the ImageNet Large Scale Visual Recognition Challenge [28], an annual public image classification competition. We used weights made publicly available through PyTorch. When discussing CIFAR-10, we trained a model ourselves adapted from the PyTorch CIFAR-10 tutorial. Unless otherwise stated, the model was trained using the cross entropy loss function. The model architecture is specified in Supplementary Methods.

In Section 3.3, we used the class of CIFAR-10 adversarially trained ResNet50 models from the Robustness library (Engstrom et al [8]). The models were trained by augmenting the input dataset about an ϵ sphere in $l2$-space about the training input and minimizing an 'adversarial' loss function which constrains the output to remain invariant over the set of perturbations lying inside this sphere. There are four models in this class, each labelled by the value of ϵ used in the training them: 0 (standard training), 0.25, 0.5 and, 1. These models are henceforth referred to as the Robustness models.

2.3 Analysis

Based on findings that layers within CNNs correlate most with stages in visual ventral stream that have roughly the same depth in the processing hierarchy [3, 17, 38], we used the penultimate layer of each network to compare against mouse V1 neurons. We made this choice for simplicity as architectures differ significantly and identifying the layer most similar to V1 in each would be a project by itself. Future work could certainly apply the analyses employed here on the penultimate layer to different layers or multiple layers.

To calculate slopes of eigenspectra, we regressed the log of the eigenvalue index number against the log of the eigenvalue itself. We frequently observed steep drop-offs in the eigenspectra beyond $10^{2.5}$, which we explained as the covariance matrices not being full rank as most models' penultimate layers had 10^3 neurons. Consequently, we calculated slopes of eigenspectra using the 10th through 316th eigenvalues, except for the eigenspectra corresponding to the outputs of Imagenet, where we calculated them using the 10th through 100th.

3 Results

3.1 CNNs Trained on ImageNet Decay Faster Than Fractal Bound

To answer the first question of whether the eigenspectra of neural activity in CNNs trained on natural images decay too slowly, placing them in the fractal representation regime, we first instantiated eleven state-of-the-art CNNs trained on natural images from the ImageNet dataset [7]. We then fed 3000 random ImageNet images into each model, extracted the activity vectors in the penultimate layer, constructed the neuron-neuron covariance matrix by averaging over the data and computed the eigenspectrum of the covariance matrix. The eigenspectra of all models is shown in Fig. 3.2.2. While the slopes vary for different architectures, they are all lower than the Stringer-Pachitariu bound, placing them in the non-fractal regime. Since all of these models are known to be vulnerable to adversarial attacks, this suggests that said vulnerability can-

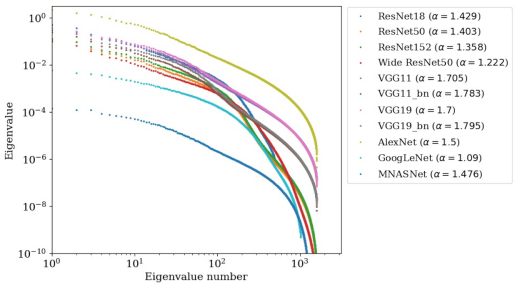

Figure 2: Eigenspectra of CNNs trained on ImageNet [7]. All CNNs have exponents below $\alpha^* \approx 1$, indicating that no model learned fractal representations.

not be explained by Stringer, Pachitariu at al.'s hypothesis that these networks have fractal neural codes.

3.2 Eigenspectra Are Invariant To Inputs Outside the Training Distribution

As we saw in Section 3.1, the eigenspectra of all the measured CNNs exist within the non-fractal regime when we used a subset of ImageNet as an input. Since the CNNs were trained on ImageNet, we expect good performance on this dataset, so it is perhaps unsurprising that the representations in the penultimate layer are smooth. Consequently, we asked what happens to the eigenspectra when the test data distribution differs from the training data distribution. In this subsection we examine the behaviour of the eigenspectra under three such variations: additive Gaussian noise, reduced dimension versions of the training dataset, and robust and non-robust versions of the training dataset.

3.2.1 Images with Additive Gaussian Noise

We added independent and identically distributed Gaussian noise of standard deviation δ (assuming images are normalized between 0 and 1) to each color component of each pixel. We tested the response of a network of each family and observed similar trends for all of them, so we only show the results for ResNet50 in Fig. 3 (left panel). The plots show that perturbing the data for the ImageNet pretrained models changes the eigenspectrum shape and slope. The characteristic shape of the ResNet50 penultimate layer activity becomes less pronounced as we add noise, since some features of the dataset disappear. The slope increases as we add more noise, and is highest for the pure noise input. For CIFAR-10, we observed a similar behavior as for ImageNet, but the changes were less pronounced. This might be due to the simplicity of the CIFAR-10 dataset. Another difference is that the inputs of pure noise had different y-intercept but very similar slope to the eigenspectra based on the dataset images with the same δ. Surprisingly, inputting only Gaussian noise produced essentially the same eigenspectra as inputting a noisy image.

Therefore, we find that if we look at the penultimate layer eigenspectra after a perturbed input, the eigenspectra do not get closer to fractal representations, but farther. This supports the conclusion from section 3.1 that models that are vulnerable to adversarial attacks have fractal representations of the space of features of the dataset.

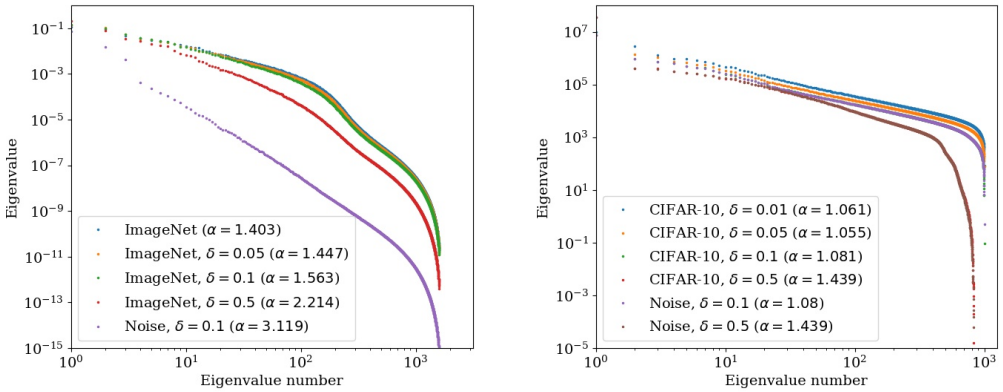

Figure 3: Eigenspectra of CNNs trained on ImageNet [7] (left) and CIFAR-10 [20] (right), with an input of additive Gaussian noise on the respective training dataset or pure Gaussian noise, where δ is the standard deviation of the noise.

3.2.2 Images of Reduced Dimension

One key aspect of Stringer, Pachitariu et al.'s experimental observations and theoretical results is that the neural code should change depending on the dimension of the input stimulus (Fig 3.2.2, Left). To test whether CNNs are similarly responsive, we constructed 7 modified versions of CIFAR-10: one whitened using zero-phase component analysis (ZCA) [2], and six projected onto the first

$1, 2, 4, 8, 16, 32$ principal component vectors. We trained a model on the original CIFAR-10 images and subsequently fed each of the seven modified versions of CIFAR-10 to the model, extracting the activity vectors and computing the eigenspectra. We observed a key difference between biological vision and artificial vision: while biological neurons change their behavior to obey the power law $\lambda_n = O(n^{-\alpha^*})$, where $\alpha^* = 1 + 2/d$, our artificial neurons were largely non-responsive to stimuli dimension (Fig 3.2.2, Right). On unmodified CIFAR-10, the model has a similar slope ($\alpha = 1.03$) to mouse V1 recordings ($\alpha = 1.04$). However, as the dimension of the input data decreased to $8, 4, 1$, mouse V1 recordings declined to $\alpha = 1.49, 1.65, 3.51$ respectively whereas the model declined to $\alpha = 1.21, 1.20, 1.20$. When presented with ZCA whitened CIFAR-10, the model changed most significantly ($\alpha = 1.51$), despite the fact that ZCA whitening decorrelates input dimensions but does not remove them. This again disagrees with mouse V1 data that displayed minimal changes ($\alpha = 1.06$) to whitened data.

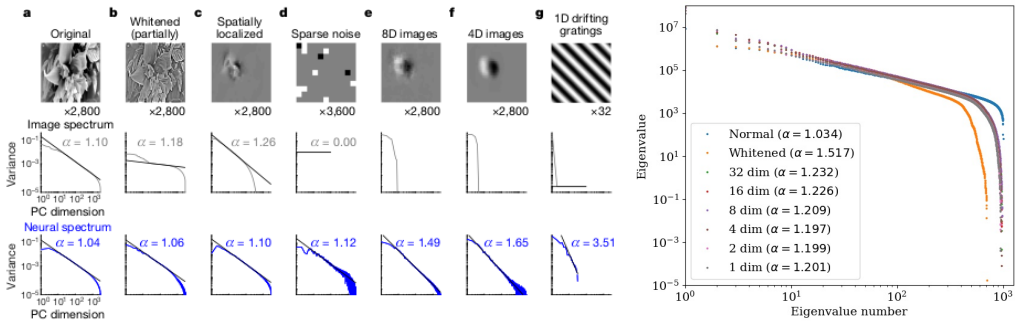

Figure 4: Left: Fig 3 from [31], showing that the slope of the neural code eigenspectrum depends on the dimension of the input stimuli d, obeying a $\lambda_n = O(n^{-1-2/d})$ power law. Right: Our experiments show that while a model trained on CIFAR-10 has a similar power law to mouse V1 recordings, the model is largely non-responsive to the dimension of the input stimuli.

We note that these results may not hold for other architectures trained on other datasets (e.g. ImageNet), especially since ResNet-50 was significantly more responsive to Gaussian input noise than our CIFAR-10 model.

3.2.3 Datasets Composed of "Robust" and "Non-Robust" Features

We examined the eigenspectra of two datasets composed of robust and non-robust features and of datasets consisting of adversarial examples (see Methods). With our CIFAR-10 model, all slopes, including the non-robust dataset obey power law decays with a slope larger than 1. This is important since it indicates that inputs consisting of non-robust features may still pass the 'smoothness' criterion that allows their output eigenspectra to decay faster than 1. Interestingly, the eigenspectra for all four datasets were identical to each other for the CIFAR-10 model, indicating that penultimate layer eigenspectra are indifferent to the feature space that the test datasets lie in. This observation may be model / architecture specific and may not be true with other models, as we see below.

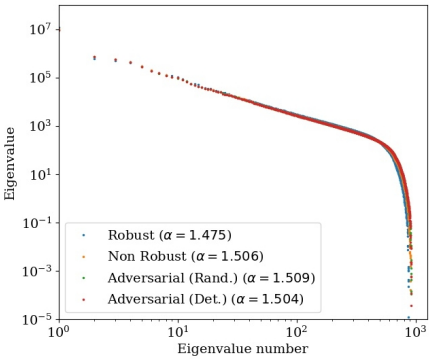

Figure 5: Penultimate layer eigenspectra of datasets consisting of adversarial examples and composed of robust and non-robust features for our CIFAR-10 model.

3.3 Different Models obey Significantly Different Power Laws

We performed the above experiments with adversarially trained models from the Robustness library [8]. The values of the principal eigenvalues of the eigenspectra are seven orders of magnitude less than those obtained with the CIFAR-10 model, and within one order of magnitude of those obtained with the ResNet50 eigenspectra for the ImageNet dataset (see Figure 3), indicating that the architecture used determines the magnitude of the eigenvalues more than variations in the test dataset. The Robustness models' penultimate layer eigenspectra for all test datasets decay significantly faster than in our CIFAR-10 model, with the slopes lying in the vicinity of 4. The steep decay in the slope is also present in the $\epsilon = 0$ model, that corresponds to the standard training case. This indicates that the difference in slopes does not arise solely from the training dataset augmentation performed while training the adversarial models, but also, in part from the difference in architectures. The behaviour of these models is consistent with the behaviour of our CIFAR-10 model, with regard to the overall invariance in the slopes obtained with different choices of the test dataset. Interestingly, unlike in the case of the CIFAR-10 model, the Robustness models exhibit slightly different eigenspectra decays with the four Ilyas et al datasets, and it appears that this class of models is less indifferent to the features that the datasets are composed of. This is perhaps unsurprising since Ilyas et al use adversarially trained models to disentangle datasets into robust and non-robust features in the first place. The behaviour exhibited by the robust dataset also differs between the four models. This observation indicates that adversarial robustness affects the representation of robust features as well.

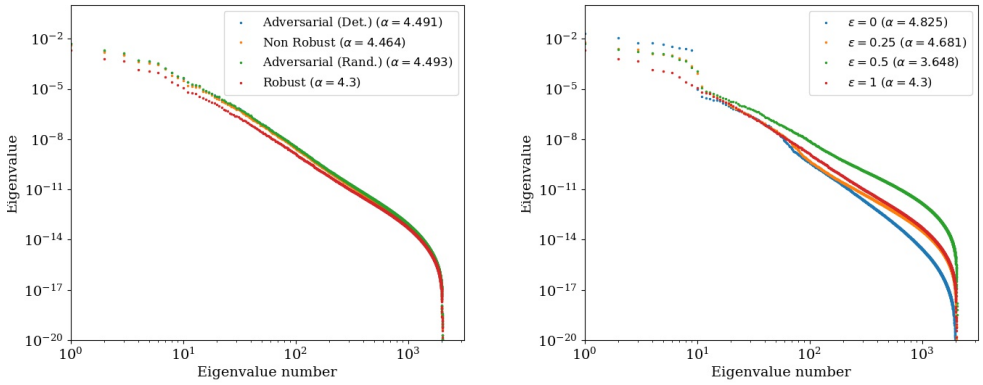

Figure 6: *Left:* Penultimate layer eigenspectra of a single Robustness model ($\epsilon = 1$) with four dataset variants. *Right:* Eigenspectra of the robust-features dataset with all four Robustness models.

3.4 Regularized Loss Functions Affect Eigenspectra

Our final direction was to explore whether regularization of the loss function could create a Stringer-Pachitariu power-law decay, and whether this representation improves resilience to adversarial examples. Let $x \in \mathbb{R}^{3\times32\times32}$ be the input CIFAR-10 image, $y \in \mathbb{R}^{10}$ be the correct CIFAR-10 class, $f(x) \in \mathbb{R}^{10}$ be the predicted CIFAR-10 class where $f(\cdot)$ is the CNN, $h_l(x) \in \mathbb{R}^{1000}$ be the activity of the penultimate layer, and c_1, c_2 be two scalar coefficients. We consider a modified loss function of the following form:

$$L(x, y) = -\sum_{i=1}^{10} y_i \log f_i(x) + c_1 || \operatorname{offdiag} h_l(x) h_l(x)^T ||_F + c_2 ||\nabla_x f(x)||_F \tag{1}$$

where $|| \cdot ||_F$ is the Frobenius norm. Each term expresses a desired property the neural code. The first is the supervised loss (cross entropy), incentivizing the neural code to encode task-necessary information. The second term penalizes covariance between neurons, pressuring the neural code to have a high-dimensional representation, while the third term penalizes the roughness of the neural representation, pressuring the neural code to be smooth. We call the first covariance penalization and the second Jacobian penalization. Prior work has considered each term (e.g. Jacobian [11, 30, 14, 15],

off-diagonal covariance [27]), although to the best of our knowledge, we are the first to consider both in the context of adversarial attacks. Here, we consider one of the most common attacks: Projected Gradient Descent (PGD) [21, 24], an iterative version of the Fast Gradient Sign Method (FGSM) [10]. FGSM takes an input x and creates an ϵ-adversarial attack x_ϵ^{adv}:

$$x^{adv} \leftarrow x + \epsilon \,\text{sign}(\nabla_x L(x, y))$$

PGD is the same algorithm repeated T times, where T is a scalar hyperparameter.

$$x_T^{adv} \leftarrow x + \sum_{t=0}^{T-1} \epsilon \,\text{sign}(\nabla_{x_t^{adv}} L(x_t^{\epsilon\, adv}, y))$$

We asked what effect c_1, c_2 have on the eigenspectra, and second, what effect c_1, c_2 have on classification accuracy against adversarial attacks. For covariance penalization $c_1 \in \{0, 0.01, 0.05, 0.1\}$ and Jacobian penalization $c_2 \in \{0, 0.5, 1, 3\}$, we trained the same CIFAR-10 model to 92% accuracy. We then generated 3000 adversarial images using PGD with the following parameters and measured each model's classification accuracy. We expected that higher values of covariance penalization c_1 would increase α, which we experimentally confirmed (Fig 7, left). Across all models, penalizing off-diagonal covariance decreased α, pushing the representations into the fractal regime; we also observed that the y-axis shifted down.

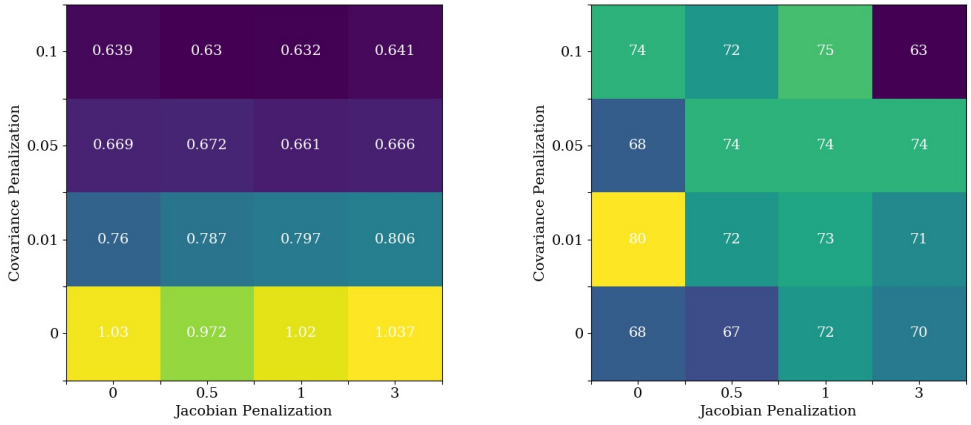

Figure 7: Models trained under Eq. (1). *Left*: α, the eigenspectrum decay exponent. Covariance penalization decreases α, indicating fractal representations. *Right*: classification accuracy against 3000 adversarial attacks generated using PGD. Jacobian penalization appears to have no relationship with accuracy.

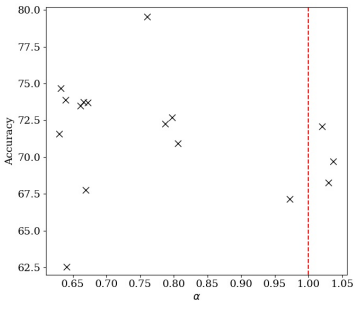

Figure 8: Classification accuracy on 3000 adversarial images versus eigenspectra exponent α. Points to the left of the red line $\alpha^* \approx 1$ indicate fractal neural codes.

We also expected that higher values of Jacobian penalization would increase classification accuracy against adversarial attacks. Surprisingly, the model with the highest accuracy against adversarial attacks had no Jacobian penalization and covariance penalization of 0.01 (Fig 7, right). We could see no relationships between Jacobian penalization and classification accuracy, noting that fractal representations have both higher and lower adversarial accuracy than non-fractal representations (Fig 3.4). This further supports the notion that adversarial vulnerability is not an emergent consequence of a fractal representation. However, we are not confident in this result because of a critical implementation detail: due to how automatic differentiation frameworks work, we were forced to penalize the norm of the column sums of the Jacobian rather than the norm of the Jacobian. This is because automatic differentiation naturally computes either Jacobian-vector and vector-Jacobian products, meaning computing the Jacobian once scales linearly with the output dimension, a cost-prohibitive operation for networks with 1000-dimensional penultimate layers.

4 Conclusion

We measured the covariance eigenspectra of the penultimate layer activations of CNNs under different inputs. In the overwhelming majority of cases, the eigenspectra decayed faster than α^*, indicating these networks learned non-fractal representations. Since these networks are known to be vulnerable to adversarial attacks, this demonstrates that a slope of $\alpha < \alpha^*$ is not a sufficient condition to make networks robust. This conclusion is further supported by our results comparing α against classification accuracy on adversarial examples, that show that even fractal α values can be more robust than non-fractal α values. Due to the network architectures and activation functions, we expect the penultimate layer representation to be piecewise differentiable. Based on Theorem 5 of Stringer, Pachitariu et al., we expect the non-fractal bound α^* to be trivially satisfied for all CNNs, so we need additional criteria to guarantee robustness in the face of adversarial attacks.

Our experiments also show that the penultimate layer covariance eigenspectrum of CNNs depends much more on the architecture, the training dataset, and the loss function than on the input data. We observed little change in the eigenspectra in response to varying inputs. Additive Gaussian noise, changes in dimensionality, whitening, and feature separation barely affected the slope α, especially for networks trained on CIFAR-10. This is inconsistent with the results derived from mouse V1 recordings by Stringer et al. [31], since they observed that varying the dimension d of the input data changed the output to follow a slope of $\alpha = 1 + 2/d$. Therefore, we see that CNNs, in contrast to mouse neurons, are not dynamically changing their behavior contingent on the input.

Finally, our results show that penalizing the norm of the end-to-end Jacobian has little effect on the decay exponent α or on classification accuracy on adversarial images. Most perplexingly, we expected that all decay exponents would satisfy $\alpha < \alpha^*$, but we empirically saw that penalizing the off-diagonal covariance components consistently drives the network's representation to the fractal region, even when the robustness to adversarial attacks is reasonably high. This suggests that the expected derivative magnitude of the network is not finite, a puzzling finding as we believed that this condition would typically be met for the architectures under consideration. Further work will be required to discover regularization that incentivizes smooth representations. One possible direction is to investigate the effects of a finite size covariance matrix on the asymptotic decays of the eigenspectrum.

Another interesting question to probe is the manner in which networks perceive or differentiate between robust and non-robust features. This is motivated by our observations that the Robustness models possess different eigenspectra for datasets composed of robust and non-robust features. Future work should additionally explore stricter conditions on the penultimate layer representation that affect the vulnerability to adversarial attacks.

5 Acknowledgments

We thank Professor Cengiz Pehlevan for his guidance and feedback on this project, and for leading such an enjoyable course this semester. We also thank Blake Bordelon for his helpful comments and suggestions.

6 References

[1] Laurent Amsaleg et al. "The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality". In: *2017 IEEE Workshop on Information Forensics and Security (WIFS)*. IEEE. 2017, pp. 1–6.

[2] Anthony J Bell and Terrence J Sejnowski. "The "independent components" of natural scenes are edge filters". In: *Vision research* 37.23 (1997), pp. 3327–3338.

[3] Charles F Cadieu et al. "Deep neural networks rival the representation of primate IT cortex for core visual object recognition". In: *PLoS computational biology* 10.12 (2014), e1003963.

[4] Zachary Charles, Harrison Rosenberg, and Dimitris Papailiopoulos. "A geometric perspective on the transferability of adversarial directions". In: *arXiv preprint arXiv:1811.03531* (2018).

[5] Yubei Chen, Dylan Paiton, and Bruno Olshausen. "The sparse manifold transform". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10513–10524.

[6] SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. "Classification and geometry of general perceptual manifolds". In: *Physical Review X* 8.3 (2018), p. 031003.

[7] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[8] Logan Engstrom et al. *Robustness (Python Library)*. 2019. URL: `https://github.com/MadryLab/robustness`.

[9] Alhussein Fawzi et al. "Empirical study of the topology and geometry of deep networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3762–3770.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. arXiv: `1412.6572 [stat.ML]`.

[11] Shixiang Gu and Luca Rigazio. *Towards Deep Neural Network Architectures Robust to Adversarial Examples*. 2014. arXiv: `1412.5068 [cs.LG]`.

[12] Umut Güçlü and Marcel AJ van Gerven. "Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream". In: *Journal of Neuroscience* 35.27 (2015), pp. 10005–10014.

[13] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] Judy Hoffman, Daniel A Roberts, and Sho Yaida. "Robust Learning with Jacobian Regularization". In: *arXiv preprint arXiv:1908.02729* (2019).

[15] Alexander G. Ororbia II, C. Lee Giles, and Daniel Kifer. *Unifying Adversarial Training Algorithms with Flexible Deep Data Gradient Regularization*. 2016. arXiv: `1601.07213 [cs.LG]`.

[16] Andrew Ilyas et al. "Adversarial examples are not bugs, they are features". In: *arXiv preprint arXiv:1905.02175* (2019).

[17] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. "Deep supervised, but not unsupervised, models may explain IT cortical representation". In: *PLoS computational biology* 10.11 (2014), e1003915.

[18] Marc Khoury and Dylan Hadfield-Menell. *On the Geometry of Adversarial Examples*. 2018. arXiv: `1811.00525 [cs.LG]`.

[19] Alex Krizhevsky. *One weird trick for parallelizing convolutional neural networks*. 2014. arXiv: `1404.5997 [cs.NE]`.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. *Adversarial examples in the physical world*. 2016. arXiv: `1607.02533 [cs.CV]`.

[22] William Lotter, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *arXiv preprint arXiv:1605.08104* (2016).

[23] Xingjun Ma et al. "Characterizing adversarial subspaces using local intrinsic dimensionality". In: *arXiv preprint arXiv:1801.02613* (2018).

[24] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[25] Lane McIntosh et al. “Deep learning models of the retinal response to natural scenes”. In: *Advances in neural information processing systems*. 2016, pp. 1369–1377.

[26] Adam Paszke et al. “Automatic Differentiation in PyTorch”. In: *NeurIPS Autodiff Workshop*. 2017.

[27] Cengiz Pehlevan and Dmitri B. Chklovskii. “Optimization theory of Hebbian/anti-Hebbian networks for PCA and whitening”. In: *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (Sept. 2015). DOI: 10.1109/allerton.2015.7447180. URL: http://dx.doi.org/10.1109/ALLERTON.2015.7447180.

[28] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[29] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[30] Jure Sokolic et al. “Robust Large Margin Deep Neural Networks”. In: *IEEE Transactions on Signal Processing* 65.16 (Aug. 2017), pp. 4265–4280. ISSN: 1941-0476. DOI: 10.1109/tsp.2017.2708039. URL: http://dx.doi.org/10.1109/TSP.2017.2708039.

[31] Carsen Stringer et al. “High-dimensional geometry of population responses in visual cortex”. In: *Nature* (2019), p. 1.

[32] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[33] Christian Szegedy et al. “Intriguing properties of neural networks”. In: (2013). arXiv: 1312.6199 [cs.CV].

[34] Mingxing Tan et al. *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. 2018. arXiv: 1807.11626 [cs.CV].

[35] Hidenori Tanaka et al. “From deep learning to mechanistic understanding in neuroscience: the structure of retinal prediction”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 8535–8545.

[36] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. “The NumPy array: a structure for efficient numerical computation”. In: *Computing in Science & Engineering* 13.2 (2011), p. 22.

[37] Rey Reza Wiyatno et al. “Adversarial Examples in Modern Machine Learning: A Review”. In: *arXiv preprint arXiv:1911.05268* (2019).

[38] Daniel LK Yamins et al. “Performance-optimized hierarchical models predict neural responses in higher visual cortex”. In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8619–8624.

7 Supplementary

7.1 Methods

7.2 Code

All code was implemented in the Python programming language, version 3.6.9. To create and train our models, we used PyTorch, an open-sourced machine learning Python library [26], version 1.2.0+cpu. Most analysis was done using NumPy, a scientific computing Python library [36], version 1.17.2.

The architecture we used for our CIFAR-10 experiments is below:

1. ReLU(Conv2d(in channels = 3, out channels = 6, kernel size = 5-by-5, strides = 1-by-1))
2. MaxPool2d(kernel size = 2-by-2, strides = 2-by-2)
3. ReLU(Conv2d(in channels = 6, out channels = 16, kernel size = 5-by-5, strides = 1-by-1))
4. MaxPool2d(kernel size = 2-by-2, strides = 2-by-2)
5. ReLU(Fully Connected Layer(in dimension = 400, out dimension = 2000))
6. ReLU(Fully Connected Layer(in dimension = 2000, out dimension = 1000))
7. Fully Connected Layer(in dimension = 1000, out dimension = 10)

7.3 Stringer, Pachitariu Theorem 5

If the expected gradient magnitude of Φ is finite i.e. $\mathbb{E}_s\left[||\nabla_s \Phi(s)||^2\right] < \infty$, then $\lambda_n = o(n^{-1-2/d})$.