Internal Representations of Vision Models Through the Lens of Frames on Data Manifolds

Henry Kvinge* HENRY.KVINGE@PNNL.GOV
Grayson Jorgenson
Davis Brown
Charles Godfrey
Tegan Emerson†
Pacific Northwest National Laboratory, Seattle, USA

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

While the last five years have seen considerable progress in understanding the internal representations of deep learning models, many questions remain. This is especially true when trying to understand the impact of model design choices, such as model architecture or training algorithm, on hidden representation geometry and dynamics. In this work we present a new approach to studying such representations inspired by the idea of a frame on the tangent bundle of a manifold. Our construction, which we call a *neural frame*, is formed by assembling a set of vectors representing specific types of perturbations of a data point, for example infinitesimal augmentations, noise perturbations, or perturbations produced by a generative model, and studying how these change as they pass through a network. Using neural frames, we make observations about the way that models process, layer-by-layer, specific modes of variation within a small neighborhood of a datapoint. Our results provide new perspectives on a number of phenomena, such as the manner in which training with augmentation produces model invariance or the proposed trade-off between adversarial training and model generalization.

Keywords: Data manifolds, hidden representations in deep learning, vision model augmentations, tangent vectors

1. Introduction

The community has made considerable progress prying open the black-box of deep learning. This has led to partial illumination of the mechanics by which a model can distill input into semantically meaningful high-level features that lead to robust predictions. Understanding this process is important since it is one way of explaining how these models perform at levels that rival human experts. Because both the input data (e.g., images) and the hidden representations within a model tend to be high-dimensional, progress has largely been built on the back of tools that leverage the geometric structure characterizing these spaces even when the spaces themselves cannot be visualized by a human.

Yet, because of the richness of deep learning representations and the high-dimensional spaces they inhabit, existing techniques by necessity provide an incomplete picture of the full relationship between model, training, data, and representation. In particular, we note

* H.K. holds a joint appointment at the University of Washington
† T.E. holds joint appointments at Colorado State University and the University of Texas, El Paso

that while many tools study the large-scale structure of representations (e.g., representation topology (Naitzat et al., 2020; Rieck et al., 2018), mutual information (Shwartz-Ziv and Tishby, 2017), model identifiability (Roeder et al., 2021)), there are fewer tools that focus on model behavior in small neighborhoods around a datapoint. Motivated by this, we describe a new tool to illuminate deep learning representations at the local level (exceptions include work focusing on adversarial examples and robustness and (Wang and Ponce, 2021) and (Zavatone-Veth et al., 2023)). To do this we leverage the notion of a frame from differential geometry. A k-frame is a choice of k linearly independent vectors from the tangent space that smoothly vary from point to point on an m-dimensional manifold M. The span of these vectors defines a subbundle of the tangent bundle of M. By applying the first ℓ-layers of a deep learning model to a k-frame, we can see how a model deforms, compresses, or expands specific directions in the immediate neighborhood around a datapoint, providing useful information about how representations of that datapoint change at the local level. We call the resulting structure a *neural k-frame* (since it will generally not be a true frame).

Neural frames have several properties that make them a valuable tool which is complementary to other methods of studying neural representations. The first is that through the choice of the input k-frame, one can study the ways in which the representation of an input example changes with respect to specific modes of variation. In this work for example, we explore frames that capture the directions of infinitesimal image augmentations, frames that point in the direction of noise, and frames generated by a diffusion model. As we show in Section 3, these different frames are processed in radically different ways by a model even when they all sit at the same datapoint. Secondly, neural frames are data efficient, only requiring a single datapoint. This is in contrast to other methods that focus on large-scale structure and hence require a whole dataset for calculation. Finally, neural frames are an intuitive and flexible construction that can often be integrated into existing tools. For example, in Section B.1 in the Appendix we show how neural frames can be combined with centered kernel alignment (CKA) to create a method of comparing the local properties of two different representations of a single datapoint.

As a proof of concept, we construct several different flavors of frames for image datasets and then apply them to a range of models with varying architectures and training methods. From these preliminary studies we are able to make a number of observations about small-scale neural representation geometry. (i) Training a model with augmentation causes it to preserve neural frames generated by small augmentations, contradicting intuition that such models would learn to collapse such modes of variation as invariance is learned. (ii) Increasing the ϵ value used in adversarial training causes a model to increasingly preserve noise directions around a datapoint at the expense more semantically meaningful augmentation directions. (iii) A model's preservation of augmentation directions correlates with its accuracy.

In summary, our contributions in this work include the following:

- We describe *neural frames*, a flexible tool which can be used to study the small-scale geometry of neural representations.
- We ground the intuitive notion of a neural frame within the theory of frames on a manifold, allowing us to connect our measurements of real models with geometry.

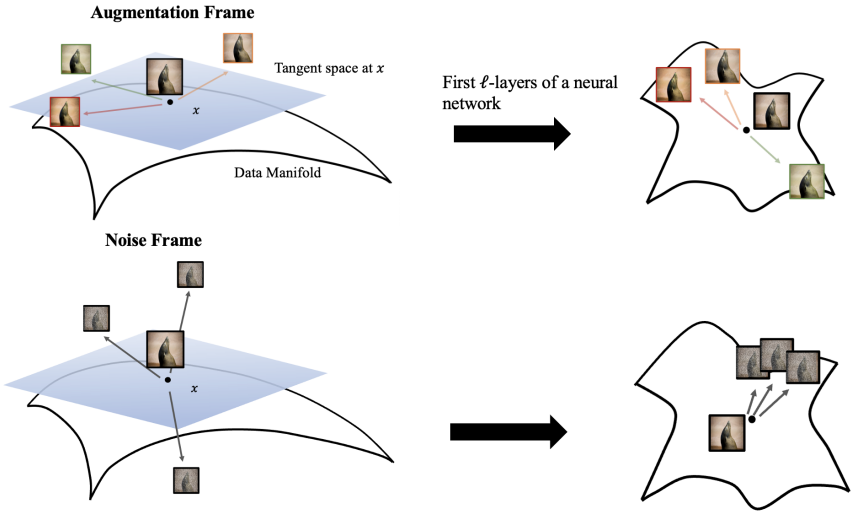

Figure 1: A cartoon visualizing an augmentation frame with three tangent vectors (derived from hue shift, brightness shift, and a small rotation and crop) and a noise frame with three noise vectors. We think of the augmentation vectors as *approximately* living in the tangent space of point x on the data manifold while the noise vectors do not. Our cartoon illustrates that models process different frames differently with the noise vectors in this cartoon being collapsed by the first ℓ layers of the network.

- We apply neural frames to a range of models with varying architectures and training histories and show that these are reflected in the local geometry of feature space.

2. Neural frames

In this section we introduce neural frames. Where possible, we center our constructions around established notions within manifold geometry since this allows us to prove a number of useful statements. A review of relevant geometric ideas such as the concept of a vector bundle, as well as some of the lemmas that support statements in this section can be found in Section H of the supplementary material.

A *k-frame* of a finite vector space V of dimension $m \geq k$ is a set of k linearly independent vectors. A *k-frame on m-dimensional manifold* M is a choice of k-frame for each tangent space T_xM which varies smoothly with respect to the structure of M. Given k linearly indepedent vector fields, there is always an open set where they form a k-frame, whose span is a subbundle of the tangent bundle. Suppose that we have a data manifold M embedded in ambient space $\mathbb{R}^n$ along with smooth functions $\mathcal{F} = \{f_1, \ldots, f_k\}$ with $f_i : (-1, 1) \times M \to M$, $f_i(t, x)$. If $f_1, \ldots, f_k$ satisfy some general conditions (see Corollary 6 in the Appendix), then we obtain a sub-vector bundle of the tangent bundle of M, $V_{\mathcal{F}}$, along with a frame $v_1(x), \ldots, v_k(x)$ obtained by differentiation of each f_i with respect to t. For the purposes

of this paper, the reader can think of each f_i as corresponding to an augmentation with t the parameter that controls the augmentation (e.g., degrees for a rotation) and x the input image.

Suppose that $F = F_\ell \circ \cdots \circ F_1 : \mathbb{R}^n \to \mathbb{R}^k$ is a neural network that decomposes into ℓ layers $F_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ with $n_0 = n$ and $n_\ell = k$. We write $F_{\leq i} := F_i \circ \cdots \circ F_1 : \mathbb{R}^n \to \mathbb{R}^{n_{i+1}}$ to denote the function that consists of the first i layers of F. While there is in general no way to use $F_{\leq i}$ to push forward a vector bundle $V_{\mathcal{F}}$ and hence a k-frame, if $F_{\leq i}$ is smooth, then its differential is a linear map $dF_{\leq i} : T_x M \to T_{F_{\leq i}(x)}\mathbb{R}^{n_{i+1}}$. This setup allows us to define neural frames.

Definition 1 *Let $v_1, \ldots, v_k$ be a k-frame on m-dimensional manifold M. Let F be a neural network as above. Then $dF_{\leq i}(v_j(x))$ is a tangent vector in $T_{F_{\leq i}(x)}\mathbb{R}^{n_{i+1}}$ and the* neural k-frame at layer i of F and point x *is the set of vectors $dF_{\leq i}(v_1(x)), \ldots, dF_{\leq i}(v_k(x))$.*

Informally, a neural k-frame is simply the object we get when we push a true k-frame through the first i-layers of a model.

Even when we have $F_{\leq i}$ and $v_j(x)$, how do we actually compute $dF_{\leq i}(v_j(x))$? This is fortunately simpler than it perhaps looks. Assuming that $v_j(x)$ is derived from a function $f_j : (-1, 1) \times M \to M$ as above, fixing $x \in M$ and letting $t \in (-1, 1)$ vary, the composition $F_{\leq i}(f(t, x))$ is a smooth path in $\mathbb{R}^{n_{i+1}}$ such that $F_{\leq i}(f(0, x)) = F_{\leq i}(x)$, and

$$dF_{\leq i}(v_j(x)) = \frac{\partial F_{\leq i}(f(t, x))}{\partial t}\Big|_{t=0}. \tag{1}$$

In practice, we compute $v_i(x) = \frac{\partial}{\partial t} f_i(t, x)|_{t=0}$ by numerically approximating the partial derivative and we compute $dF_{\leq i}(v_j(x))$ by approximating the derivative on the right hand side of equation 1.

Once we have a neural frame, $dF_{\leq i}(v_1(x)), \ldots, dF_{\leq i}(v_k(x))$, we can extract a range of statistics that help diagnose what $F_{\leq j}$ is doing to the data manifold at point x. Since we assume that $v_1(x), \ldots, v_k(x)$ belong to a sufficiently small neighborhood U of data manifold M such that U is approximately linear, an initial idea may be to look for changes in rank of the matrices $A_{x,i}$ with columns $dF_{\leq i}(v_1(x)), \ldots, dF_{\leq i}(v_k(x))$, as i varies. Unfortunately, rank is sensitive to noise and is hence unsuitable for this application. An appealing alternative is *stable rank* (Rudelson and Vershynin, 2007), which is the ratio between squared Frobenius norm and the squared spectral norm of a matrix. For $A_{x,i}$ this is

$$r(A) := \frac{||A_{x,i}||^2_{\text{frob}}}{||A_{x,i}||^2_{\text{spec}}}.$$

It can easily be calculated by computing the singular values $\sigma_1, \ldots, \sigma_k$ of $A_{x,i}$ and then

$$r(A) = \frac{1}{\max_i(\sigma_i)} \sum_i \sigma_i^2.$$

This definition shows that stable rank captures the extent to which data variation is captured in a small number of dimensions. Since k is small in practice (< 50), computing stable rank via a singular value decomposition of $A_{x,i}$ is quick. We note that stable rank has found a number of useful applications to deep learning (Sanyal et al., 2019).

Stable rank is a lower bound on rank, and we can interpret a decrease (respectively, increase) in stable rank when moving from $v_1(x), \ldots, v_k(x)$ to $dF_{\leq i}(v_1(x)), \ldots, dF_{\leq i}(v_1(x))$ to mean that $F_{\leq j}$ compresses (resp. expands) the data manifold M in the directions captured by $v_1(x), \ldots, v_k(x)$. This of course does not mean that $F_{\leq j}$ compresses all of M at x since in general $v_1(x), \ldots, v_k(x)$ will only span a small subspace of the tangent space of M at x. Nevertheless we will see that some of results we obtain using stable rank reflect patterns seen in past intrinsic dimension experiments.

We end this section by discussing two different types of frames which seem particularly interesting.

Augmentation frames: This neural frame is generated by image augmentations $f_i : (a, c) \times M \to M$ with the following properties: (1) as implied by the domain and range of f_i above, f_i transforms one natural image (on M) to another natural image (on M). While this latter image was not actually captured by a camera (instead being produced in software) it should be plausible that it could have been. (2) Aside from the input image, f_i is also controlled by a parameter $t \in (a, c)$ for $a < c \in \mathbb{R}$ such that for some $b \in (a, c)$, $f_i(b, x) = x$ for all $x \in M$. For example, if $f_{\text{rot}} : (-180, 180) \times M \to M$ is image rotation, with the first parameter measuring the number of degrees that an image will be rotated, then it is always the case that $f_{\text{rot}}(0, x) = x$.

The frame $v_1, \ldots, v_k$ derived from $f_1, \ldots, f_k$ describes a number of pseudo-naturalistic directions in which an image can vary without leaving the image manifold. The neural frame associated with this frame tells us how a model handles change in these directions locally. In Table 1 in the Appendix, we list the image augmentations that we used, the library we used to implement them, and the augmentation parameters that were used in our experiments.

Some image augmentations come with more than a single real parameter that a user can choose from. For example, when rotating an image, one can often pick the pixel coordinates of the point which will be the folcrum of the rotation (for example, in (Marcel and Rodriguez, 2010)). How many versions of the augmentation should one add to the augmentation frame in such cases? In a 224×224 image there are 50176 pixels that we could rotate around. How many can be added before the corresponding tangent vectors become linearly dependent? In cases where the underlying augmentation corresponds to the action of a Lie group (including this case, where the Lie group is the special Euclidean group $SE(2)$ which is generated by all translations and rotations of the plane), Lie theory can provide an answer. We begin by recalling that the action of a Lie group G on a manifold M induces a linear map from the Lie algebra $\mathfrak{g}$ to T_xM for any $x \in M$.

Proposition 2 *(Theorem 20.15 (Lee, 2013)) Let $\mathfrak{g} = T_0G$ be the Lie algebra of G and $\rho : G \times M \to M$ a Lie group action of G on M. Suppose $x \in M$, and define $\text{ev}_x : G \to M$ as $\text{ev}_x(g) = \rho(g, x)$. Then ev_x is a smooth map and the differential of ev_x at the identity element $e \in G$ is a linear map $d\text{ev}_x : \mathfrak{g} \to T_xM$.*

Given Proposition 2, our problem is equivalent to identifying the dimension of the image of $d\text{ev}_x$ This will tell us the maximum number of linearly independent tangent vectors that can be generated by the action of G. To state the solution, we require a piece of terminology: the *stabilizer* of a point $x \in M$ is the subgroup $G_x = \{g \in G \,|\, gx = x\}$.

Proposition 3 *The natural map $\mathfrak{g} \to T_xM$ is injective if and only if the stabilizer G_x is discrete.*

In our rotation example the stabilizer $SE(2)_x$ of a natural image x almost always consists of the identity alone, hence is discrete. Since the dimension of Lie group $SE(2)$ is 3 and a Lie algebra's dimension (as a vector space) is equal to the manifold dimension of its corresponding Lie group, the subspace of T_xM spanned by tangent vectors generated by all possible rotations at different points in a image is 3. Thus we conclude that for most images we only need to include tangent vector approximations for rotations at 3 points in an image. In our experiments we choose to rotate at pixels $(0, 0), (50, 50)$, and $(-50, 50)$.

Diffusion frames: The recent work (Luzi et al., 2022), describes how diffusion models can be used to sample locally around an image. In essence, the method they describe, called Boomerang, adds a user chosen amount of noise to an image (driving it away from the image manifold) and then uses the diffusion model to bring it back to the image manifold (but not to the original point). In this process the image will be subtly altered in a naturalistic way. This method fits nicely within our scheme of neural frames: we use Boomerang to produce k distinct perturbations of an image (to match our augmentation frame, in our experiments $k = 19$), then we define a k-frame with these. We assume that the perturbations generated by the diffusion model are small enough so that the linear path from a perturbed image to the real image lies on the image manifold.

3. Experiments

Having developed neural frames, we show their utility by using them to probe the local behavior of deep learning models. We give full experimental details in Section J in the supplementary materials. Unless noted otherwise, we use publicly available weights from Torchvision (Marcel and Rodriguez, 2010) or Timm (Wightman, 2019). To simplify diagrams, we omit layer names providing their numerical correspondence in Tables 2-10 in the supplementary material. Unless otherwise noted, we performed our evaluation on 40 random ImageNet training images. We did not see substantial changes in results when either using a larger sample size or using the test set[1].

We utilize four different types of frames in our experiments which we describe here. (1) *Gaussian noise:* We perturb an image with random Gaussian noise with mean and variance which we normalize to match the statistics of vectors in our augmentation frame. Note this is not a frame on the image manifold itself. (2) *Augmentation frame:* We use the augmentations listed in Table 1 to generate an augmentation frame (example images of augmentations are found in Figure 13 in the supplementary material). (3) *Random rotation of augmentation frame:* We randomly rotate the augmentation frame above so it retains its geometric structure but loses its semantic meaning. (4) *Perturbations via stable diffusion:* We use the Boomerang method (Luzi et al., 2022) to generate samples from around an ImageNet image and take these samples as perturbations to build a frame (example images of this frame are found in Figure 14).

1. The latter phenomenon suggests that model generalization may be detected by statistics even at the very local level.

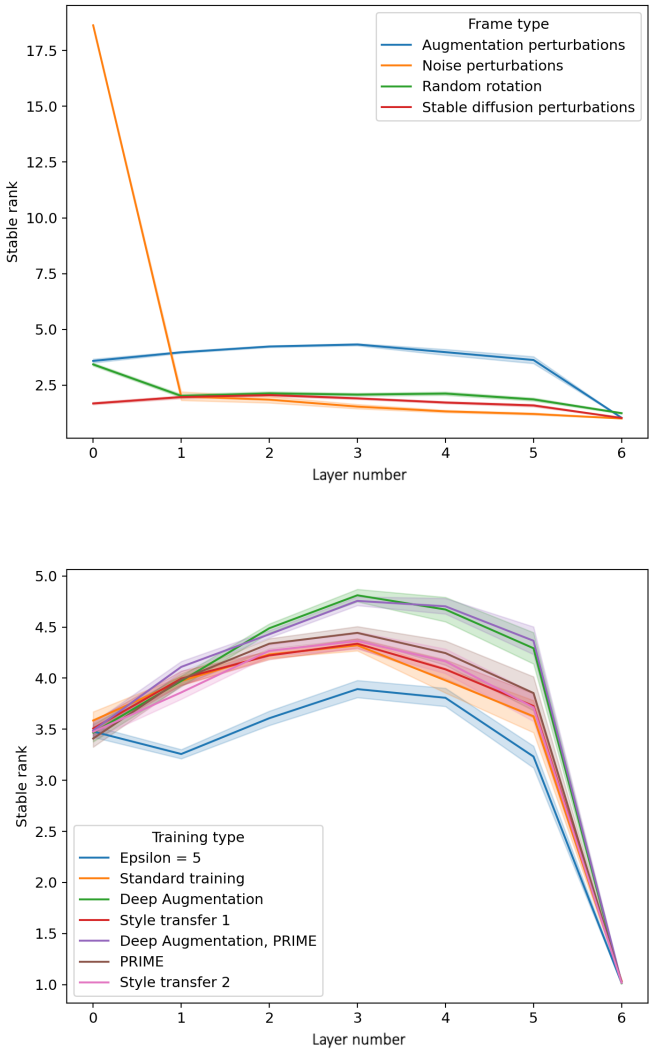

Figure 2: **(Left)**The stable rank of different types of frames measured at various layers of a ResNet50 model. **(Right)** The stable rank (by layer) of ResNet50 models evaluated with respect to augmentation frames on ImageNet images where each model was trained with a different augmentation method. Layer zero corresponds to model input and the last layer corresponds to model output (see Table 11 for the names of layers 1-5). Shaded regions indicate 95% confidence intervals over 40 randomly selected ImageNet images.

Models generally preserve on-manifold tangent vectors and collapse vectors that point off manifold: It is reasonable to ask whether a CNN or transformer actually "sees"

different frames differently. One might worry that on the small scales that we work, the different frames we construct do not capture meaningful differences in model representations. For example, can a model actually tell the difference between an augmentation frame and a noise frame?

To test this, we plot the stable rank for a number of different frames at different layers of a ResNet50 pretrained on ImageNet (Marcel and Rodriguez, 2010). The results are shown in Figure 2 (left) (Figure 8 in the supplementary material has results for the same experiment with a ViT). We can see that even with the coarse statistic of stable rank, different neural frames exhibit distinct behavior when processed by the model. The neural frame generated from Gaussian noise predictably has the highest stable rank in the ambient space (layer 0), but this drops quickly in both models as the frame is processed. This suggests that models preserve those directions more representative of natural variation at the expense of random directions. In contrast, augmentation frames, which simulate directions of natural variation of imagery are generally more preserved from layer to layer, only being collapsed in the final classification layer.

As a sanity check, one might wonder if this phenomenon is a consequence not of the directions that augmentation frames point relative to noise frame, but of other structural features of the frame itself. For example, inspection of the stable rank of the input frames in Layer 0 of Figure 2 (left) (prior to processing by the model) show that the noise frame is close to being an orthogonal set of vectors while the augmentation frame has significant linear redundancies. To explore this, we randomly rotate the augmentation frame so it no longer points in the direction of natural changes to the image but keeps other structural features. When we do this we see that this rotated frame, like the noise frame, is collapsed by the model. This provides strong evidence that, even at the very smallest scales, models recognize and preserve directions that simulate natural variation found in imagery.

Training with augmentation causes models to better preserve on-manifold augmentation frames: In order to better understand the impact of training with augmentation at the local level, we explored the stable rank of augmentation frames for a range of models trained with (and without) different types of heavy augmentation. We consider ResNet50 models trained with the augmentation methods PRIME (Modas et al., 2021), Deep Augmentation (Hendrycks et al., 2021), and Stylized ImageNet (Geirhos et al., 2018). Our results are shown in Figure 2 (right) where we see that generally, models trained with extra augmentation have neural frames with higher stable rank, indicating that these models more faithfully preserve (and to some extent even expand) frames represented by small augmentations. Note that this may seem unexpected given that training with augmentation is generally done to build invariance to natural variation in images. This might lead one to conclude that training with augmentation should cause augmentation frames to collapse as a model consolidates different augmented versions of the same data point. Figure 2 (right) suggests that this must happen only in the final layers of a model and that instead, in earlier layers training with augmentation causes a model to learn more distinct and structured representations of different augmentations of a single input. This speculation agrees with observations found in (Kvinge et al., 2022).

On the other hand, adversarial training degrades the preservation of augmentation frames but improves the preservation of noise frames: It has been empirically confirmed via a range of different methods that adversarial training has effects on the way

that computer vision models process data at the local level (Engstrom et al., 2019). To investigate whether this can be seen at the level of neural frames, we calculated the stable rank for 5 layers of several different ResNet50 models, each trained with a different l_2-robust ϵ bound of adversarial training with weights from (Salman et al., 2020).

On the left in Figure 3 we show the stable rank over 40 random ImageNet images with respect to the augmentation frame and models with various strengths of adversarial training (here ϵ gives the ℓ_2 bound on adversarial examples shown to the model during training). We observe that the stable rank of our augmentation frames generally decreases slightly as the strength of adversarial training increases. Furthermore, these differences are most pronounced at earlier layers of the model. On the other hand, we can see that when we substitute the augmentation neural frame for the off-manifold noise neural frame (right, Figure 3) that the opposite pattern holds and stable rank generally increases as the strength of adversarial training increases.

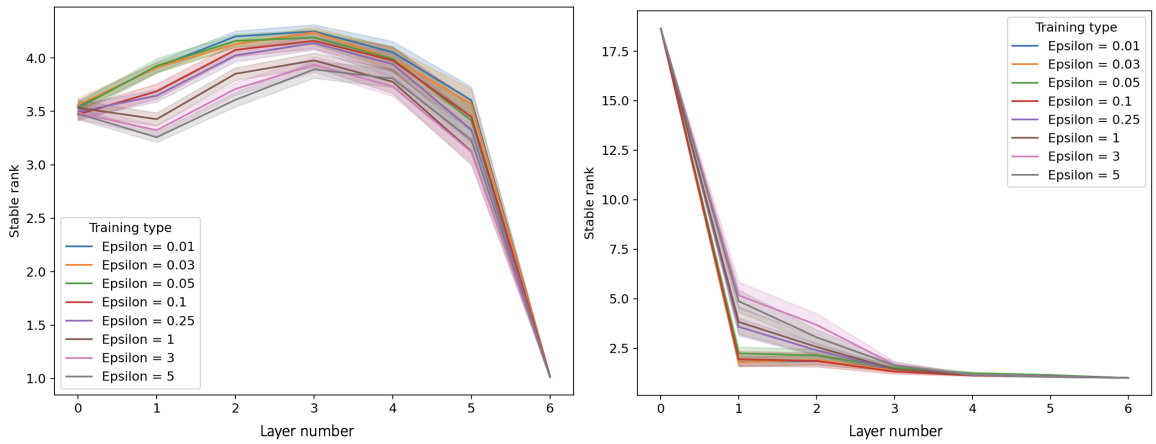

Figure 3: **(Left)** The stable rank (by layer) of adversarially trained l_2-robust ResNet50 models with varying ϵ values evaluated with respect to augmentation frames on ImageNet images. Layer zero corresponds to model input and the last layer corresponds to model output (see Table 11 for the names of layers 1-5). **(Right)** The same models evaluated on noise frames. Shaded regions indicate 95% confidence intervals over 40 randomly selected ImageNet images.

The average stable rank of augmentation frames is correlated with model accuracy: It is natural to ask whether statistics associated with neural frames have any relationship with other characteristics of a model. In Figure 4, we show that higher average stable rank of augmentation frames is correlated with model accuracy. This observation fits well with our speculation above that preservation of augmentation frames (as measured by stable rank) may be tied to a model's fit to the underlying image manifold.

We end with a couple final observations that we explore more thoroughly in the supplementary material: (i) Neural frames reveal that over the course of training, models initially locally compress the image manifold and then gradually expand it as training progresses

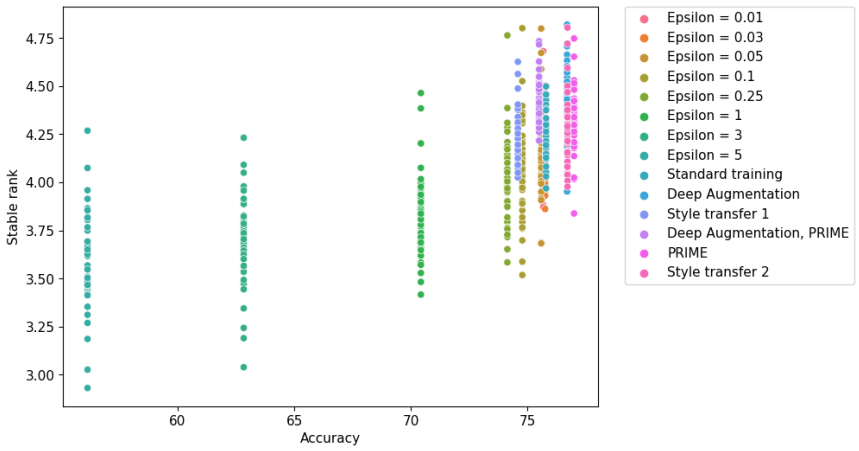

Figure 4: The stable rank versus accuracy for a range of ResNet50 models trained on ImageNet. Each point represents the stable rank of a frame on a single image/model pair.

(see Figure 9 in the supplementary material). It would be interesting to tie this to previously observed phenomena e.g., the information bottleneck (Tishby and Zaslavsky, 2015). (ii) The stable rank of augmentation neural frames tend to vary less across CNN models when compared to transformers, suggesting that transformers compress and stretch data manifolds more during processing (see Figures 11-12). (iii) We describe frame CKA, which can reveal inter-layer differences in ViTs that traditional CKA struggles to capture.

4. Conclusion

While data manifolds play a central role in our understanding of how and why deep learning works, extracting any tangible information about them is challenging. In this paper we provide a new tool, neural frames, to help probe the ways that deep learning models interact with data manifolds. We show that neural frames vary substantially for models that were trained in different ways, providing some insight into how choices in architecture and training method impact the way that a model processes data.

Acknowledgments

This research was supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative at Pacific Northwest National Laboratory. It was conducted under the Laboratory Directed Research and Development (LDRD) Program at at Pacific Northwest National Laboratory (PNNL), a multiprogram National Laboratory operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC05-76RL01830.

References

Laurent Amsaleg, James Bailey, Dominique Barbe, Sarah Erfani, Michael E Houle, Vinh Nguyen, and Miloš Radovanović. The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2017.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Serguei Barannikov, Ilya Trofimov, Nikita Balabin, and Evgeny Burnaev. Representation topology divergence: A method for comparing neural network representations. *arXiv preprint arXiv:2201.00058*, 2021.

Rajendra Bhatia. *Matrix Analysis*. Springer Science & Business Media, November 1996. ISBN 978-0-387-94846-1.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1): 1–8, 2017.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Haydn T Jones, Jacob M Springer, Garrett T Kenyon, and Juston S Moore. If you've trained one you've trained them all: inter-architecture similarity increases with robustness. In *Uncertainty in Artificial Intelligence*, pages 928–937. PMLR, 2022.

Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

Alexander Kirillov, Jr. *An Introduction to Lie Groups and Lie Algebras*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2008. doi: 10.1017/CBO9780511755156.

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *arXiv preprint arXiv:2305.06329*, 2023.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

Henry Kvinge, Tegan Emerson, Grayson Jorgenson, Scott Vasquez, Tim Doster, and Jesse Lew. In what ways are deep neural networks invariant and how should we measure this? *Advances in Neural Information Processing Systems*, 35:32816–32829, 2022.

John M Lee. Smooth manifolds. In *Introduction to smooth manifolds*, pages 1–31. Springer, 2013.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

Elizaveta Levina and Peter J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, page 777–784, Cambridge, MA, USA, 2004. MIT Press.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

Lorenzo Luzi, Ali Siahkoohi, Paul M Mayer, Josue Casco-Rodriguez, and Richard Baraniuk. Boomerang: Local sampling on image manifolds using diffusion models. *arXiv preprint arXiv:2210.12100*, 2022.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

V. A. Marčenko and L. A. Pastur. Distribution of Eigenvalues for Some Sets of Random Matrices. *Sbornik: Mathematics*, 1(4):457–483, April 1967. doi: 10.1070/SM1967v001n04ABEH001994.

Apostolos Modas, Rahul Rade, Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Prime: A few primitives can boost robustness to common corruptions. *arXiv preprint arXiv:2112.13547*, 2021.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *Journal of Machine Learning Research*, 21(184):1–40, 2020. URL http://jmlr.org/papers/v21/20-345.html.

Chris Olah. Visualizing representations: Deep learning and human beings, 2015. URL http://colah.github.io/posts/2015-01-Visualizing-Representations/. Accessed on May 16, 2023.

Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2071–2081, 2022.

Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. *arXiv preprint arXiv:2104.08894*, 2021.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. *arXiv preprint arXiv:1812.09764*, 2018.

Geoffrey Roeder, Luke Metz, and Durk Kingma. On linear identifiability of learned representations. In *International Conference on Machine Learning*, pages 9030–9039. PMLR, 2021.

Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21–es, 2007.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

Amartya Sanyal, Philip HS Torr, and Puneet K Dokania. Stable rank normalization for improved generalization in neural networks and gans. *arXiv preprint arXiv:1906.04659*, 2019.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

D.J.A. Trotman. Stability of transversality to a stratification implies whitney (a)-regularity. *Inventiones mathematicae*, 50:273–278, 1978/79. URL http://eudml.org/doc/142616.

Binxu Wang and Carlos R Ponce. The geometry of deep generative image models and its applications. *arXiv preprint arXiv:2101.06006*, 2021.

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

Jacob A Zavatone-Veth, Sheng Yang, Julian A Rubinfien, and Cengiz Pehlevan. Neural networks learn to magnify areas near decision boundaries. *arXiv preprint arXiv:2301.11375*, 2023.

Appendix A. Limitations

While neural frames provide unique and valuable information about how a network processes a data manifold locally, this information is never the full story. For example, in most cases the frames we use span proper subspaces of the tangent space. Thus, there may be changes to the tangent space that we miss because they are orthogonal or nearly orthogonal to all vectors in the frame that we use. Further, augmentation frames may be challenging to use in specialized scenarios where augmentations that preserve the particular image manifold under consideration are not a priori known.

Appendix B. Related work

Mathematical tools to study the internal representations of deep learning models: Given deep learning's remarkable performance on a broad range of tasks, substantial research effort has gone into better understanding how and why it works. While these studies have taken a range of forms, understanding the way that models represent input data throughout their layers has been a prominent theme. Because of the high-dimensionality of such representations, mathematical tools are generally needed in order for a human to analyze or visualize them. While the standard suite of dimensionality reduction methods are used for visualization (Olah, 2015), these often fail to be sufficiently quantitative for analysis. Instead, tools from geometry, topology, and probability theory have found application as these fields routinely study spaces that are challenging for humans to understand using visual intuition. Along these lines, several works have explored neural representations in terms of a range of mathematical descriptors, from properties derived from algebraic topology (Naitzat et al., 2020; Barannikov et al., 2021), to intrinsic dimension (Ansuini et al., 2019; Pope et al., 2021; Ma et al., 2018), to mutual information (Shwartz-Ziv and Tishby, 2017). Our work complements these by focusing on the small-scale structure of representations unlike most of the works above which focus on large-scale structure (that is, at the level of multiple points in the dataset), as well as structure that targets particular modes of variation (e.g., how a model represents noise directions vs directions corresponding to augmentations).

Comparing representations: Beyond understanding them in isolation, in many cases it is useful to be able to compare the representations of the same data by different models or different layers of the same model. Notable methods include: canonical correlation analysis (CCA) (Hardoon et al., 2004), along with its variants SVCCA (Raghu et al., 2017) and PWCCA (Morcos et al., 2018) (see (Klabunde et al., 2023) for a recent survey outlining the relationship between these and other methods). Another method of comparing representations is neural stitching (Lenc and Vedaldi, 2015) which (roughly) measures whether one model can learn from another's representation. In this work we use centered kernel alignment (CKA) (Kornblith et al., 2019), one of the most popular tools for comparing internal representations of deep learning models. In contrast to all the methods mentioned here, the frame CKA approach that we propose in this paper focuses on similarity between local representations around a data point. This makes it complementary to other methods which often focus on large-scale structure between many distinct datapoints (in most cases an entire validation set).

B.1. Frame CKA

Let $D = \{x_1, \dots, x_d\}$ be a dataset in $\mathbb{R}^n$ and $F^1, F^2 : \mathbb{R}^n \to \mathbb{R}^k$ two neural networks. Let $F^1_{\leq i_1}(D)$ be the matrix whose rows are $F^1_{\leq i_1}(x_1), \dots, F^1_{\leq i_1}(x_d)$ with analogous notation for $F^2_{\leq i_2}(D)$. Using the notation from Section 2, the centered kernel alignment (CKA) (Kornblith et al., 2019) score for models F^1, F^2 at layers i_1 and i_2 respectively in (terms of D) is defined as

$$\mathrm{CKA}(F^1_{\leq i_1}(D), F^2_{\leq i_2}(D)) = \frac{||\mathrm{Cov}\big(F^1_{\leq i_1}(D), F^2_{\leq i_2}(D)\big)||_F^2}{||\mathrm{Cov}\big(F^1_{\leq i_1}(D), F^1_{\leq i_1}(D)\big)||_F ||\mathrm{Cov}\big(F^2_{\leq i_2}(D), F^2_{\leq i_2}(D)\big)||_F}$$

where Cov denotes covariance and $||\cdot||_F$ is the Frobenious norm. Very roughly, CKA measures the structural similarity of representations of datapoints D in F^1 at layer i_1 vs the representation of D in F^2 at layer i_2 with higher scores (closer 1) indicating representations are that structrally similar. Notably, CKA is invariant to orthogonal transformation, which fits with the intuition that rotating a model's representation does not meaningfully change its structure (see (Klabunde et al., 2023) for further discussion on this and other invariances in similarity metrics).

The same reasons that CKA is a useful tool for comparing high-dimensional model representations make it appropriate to comparing neural frames. In particular, if $v_1(x), \ldots, v_k(x)$ is a k-frame at $x \in \mathbb{R}^n$, then we can apply CKA to the matrices whose rows are

$$dF^1_{\leq i_1}(v_1(x)), \ldots, dF^1_{\leq i_1}(v_k(x)) \quad \text{and} \quad dF^2_{\leq i_1}(v_1(x)), \ldots, dF^2_{\leq i_1}(v_k(x))$$

respectively. We call the resulting statistic the *frame CKA score* of F^1 and F^2 at layers i_1 and i_2 for frame $v_1(x), \ldots, v_k(x)$. Following the standard interpretation of CKA scores, a frame CKA score close to 1 indicate that F^1 and F^2 represent frame $v_1(x), \ldots, v_k(x)$ similarly at layers of i_1 and i_2 respectively. Note that unlike standard CKA which compares the representation of a collection of points, frame CKA compares the arrangement of the vectors of a neural frame at a single point.

Appendix C. How does the stable rank of a frame relate to intrinsic dimension?

Given that the dimension of a manifold can be defined as the vector space dimension of its tangent space, one might ask how the stable rank of a neural frame (which in some cases is also related to the tangent space of a data manifold) relates to the intrinsic dimension of a neural representation. This is especially pertinent given the large number of works that investigate neural representations through the lens of intrinsic dimension (Ansuini et al., 2019; Pope et al., 2021; Amsaleg et al., 2017; Ma et al., 2018). In this short section we compare the stable rank of neural frames to intrinsic dimension to get a better sense of what both are capable of telling us.

- **Manifold dimension:** Intrinsic dimension is designed to estimate the dimension of the manifold underlying a dataset. Unless we are using a frame whose vectors span the entire tangent space of the manifold, the stable rank will not tell us the intrinsic dimension (though it is a lower bound).

- **Number of points required:** Intrinsic dimension generally requires many real data points to calculate, and this number increases as the actual intrinsic dimension increases. A broad range of works have tried to provide detailed estimates of the number of points necessary to get a reliable estimate (e.g., (Fefferman et al., 2016)). On the other hand, the stable rank of a frame can be calculated with a single datapoint provided one knows how to perturb that datapoint in order to generate the frame. Because there is variation between the stable rank of frames on different datapoints, we advocate using at least several datapoints and taking an average.

- **Analyzing different sources of variation:** A variety of works that have investigated the intrinsic dimension of the hidden activations of deep learning models have noted that these models tend to decrease the original dimension of the data manifold as data passes through the model (Ansuini et al., 2019). One can ask what specific sources of variation are collapsed in this process. Does a drop in intrinsic dimension from one layer to the next represent the fact that the model ignores some structured degree of freedom (e.g., color)? It is not straightforward to measure this with intrinsic dimension alone. On the other hand, since frames can capture specific directions of variation, these kinds of questions become accessible.

- **Local vs very local:** Intrinsic dimension estimators come in a variety of flavors. Some use the entire dataset to estimate dimensionality, while other more recent approaches average over many local neighborhoods. In all these cases, the size of the neighborhood used is constrained by the dataset. If the dataset is sparsely sampled (and hence points are further apart), the neighborhoods used are by necessity larger. On the other hand, since neural frames utilize various tools to perturb a datapoint, the neighborhoods they study can often be made much smaller.

- **Use of real vs synthetic data:** In most cases intrinsic dimension estimators only use real datapoints from the dataset. On the other hand, the approaches to neural frames that we describe here use augmentations or generative models to create close neighboring points that estimate tangent vectors.

Despite these differences, we find that in certain cases at least, intrinsic dimension and the stable rank of neural frames appear to capture similar patterns in neural representations. Figure 5 shows the intrinsic dimension of 5,000 ImageNet images at different layers of a vision transformer (left vertical axis) as captured by intrinsic dimension estimators MLE (Levina and Bickel, 2004) and TwoNN (Facco et al., 2017), vs the stable rank of an augmentation frame (right vertical axis). We see that while these statistics differ numerically, their curves have similar shapes.

Appendix D. How does the choice of k impact the stable rank of a frame?

In our experiments above, we mostly restricted ourselves to 19-frames as this was the total number of augmentations that we found that were suitable for use in an augmentation frame. It is worth asking what happens when we vary k in a k-neural frame. Do our conclusions remain stable? In Figure 6 we show the result of decreasing k for the augmentation frames (described in Section J.1) for a ResNet50 trained on ImageNet. We find that increasing the value of k in augmentation k-frames increases the stable rank of the corresponding neural frames. Nevertheless, the shape of the curves (e.g., layers where stable rank increases) seems to mostly remain the same after sufficiently large k. On the other hand, increasing k when using noise frames does not appreciably change the stable rank of the corresponding neural frames (though the stable rank of the input increases predictably with the number of frames).

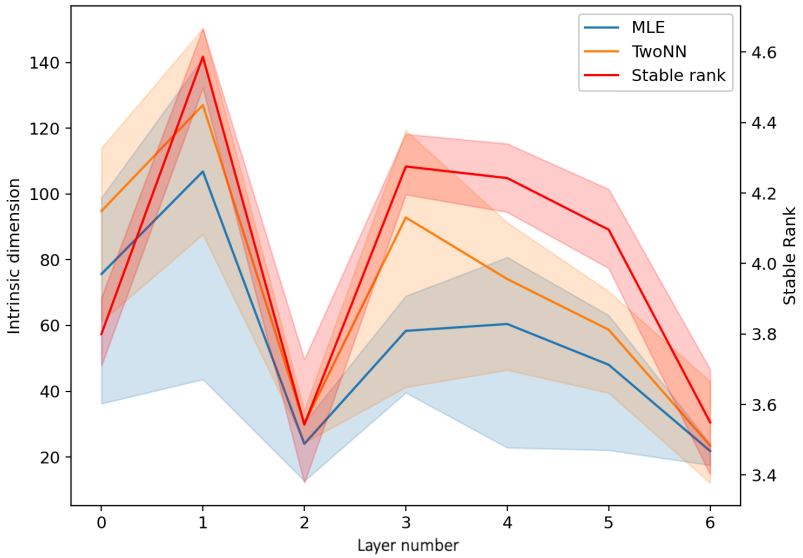

Figure 5: An estimation of the intrinsic dimension of the hidden activations of $5,000$ ImageNet images using MLE and TwoNN within a ViT model. We include the stable rank for comparison. Input and output layers are omitted, so 0 corresponds to the first hidden representation. Shaded regions indicate 95% confidence intervals over 40 randomly selected ImageNet images for stable rank and three random samplings of $5,000$ ImageNet images for MLE and TwoNN.

The fact that increasing k increases the stable rank of neural augmentation frames but does not increase the stable rank of neural noise frames reinforces the idea that for most models, directions of change associated with augmentation are individually preserved (hence adding them to the input frame causes changes to the corresponding neural frame), whereas noise directions mostly are not. In future work it would be interesting to understand how the addition of specific augmentation directions impact stable rank. Overall, it appears that qualitative patterns in stable rank per layer are mostly preserved when k is changed provided that k is sufficiently large.

Appendix E. Are different types of frames processed differently in a transformer architecture?

In Section 3 we showed that different types of frames are processed very differently by a ResNet50. One might ask if a similar statement holds for vision transformers, which have substantially different architecture. In Figure 8 we show a similar plot to that found in Figure 2. We see a similar phenomenon holds with minor differences. For example, the vision transformer tends to preserve a noise frame somewhat longer than the ResNet50

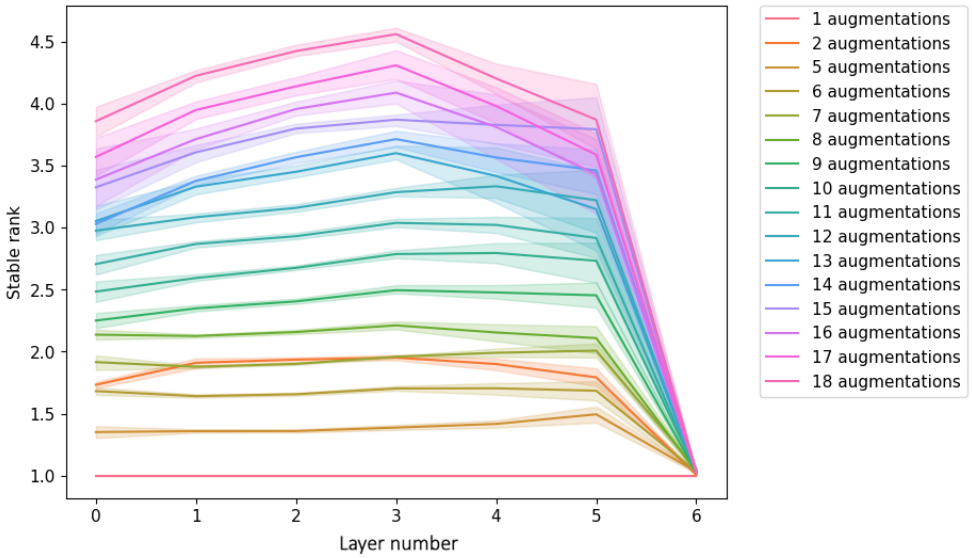

Figure 6: The stable rank (by layer) of an ImageNet trained ResNet50 (Marcel and Rodriguez, 2010) evaluated with respect to augmentation frames with varying k. The order in which specific augmentations were added was chosen randomly.

does. Given the results in Section 3, we speculate that this may relate to vision transformer's purported adversarial robustness (Paul and Chen, 2022).

Appendix F. Stable rank over the course of training

To better understand how the stable rank of a frame changes over the course of training, we saved the weights of a ResNet18 (He et al., 2016) trained from scratch on ImageNet every 10 iterations (for 1000 iterations) and then every 100 iterations for the approximately 16 remaining epochs. The training hyperparameters that we used can be found in Table 13.

In Figure 9 we show the stable rank (by layer) for this ResNet18 with respect to an augmentation frame at different stages of training. We see that at a large scale the general trend is for stable rank to increase as training increases, but that these changes are most significant in the later layers of the model. For example, the latent space layer (layer 5), increases from an initial stable rank around 1.5 to a stable rank of 3.5, an increase of 2, while the stable rank of layer 1 (in one of the first blocks of the model), only increases from 3.5 to 4, an increase of only .5. We conjecture that one effect of the later stages of training is that a model gains the tendency to preserve those frames related to natural changes of an image. This guess is supported by Figure 10 (right) which shows that while stable rank increases throughout training for frames of naturalistic directions, it decreases for noise frames whose directions lack any connection to the content of the image.

Interestingly, we find stable rank also peaks (though not as high as later) once in the early iterations of training. In Figure 10 (left) we see that stable rank increases for approximately

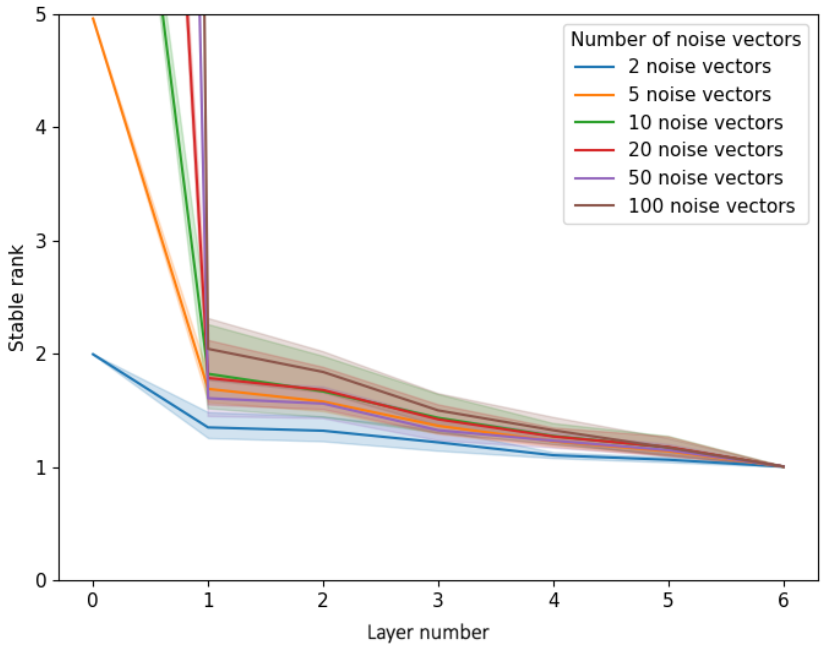

Figure 7: The stable rank (by layer) of an ImageNet trained ResNet50 (Marcel and Rodriguez, 2010) evaluated with respect to noise frames for varying k.

the first 50 iterations of training and then decreases again until around iteration 200. It then slowly increases for the rest of training. It would be interesting to understand what drives these dynamics.

Appendix G. Stable rank and architecture

In Figures 11 and 12 we plot the stable rank (as a function of layer) for an augmentation frame and two different families of architectures: CNNs and vision transformers. Shaded regions depict 95% confidence intervals calculated over 40 random ImageNet images. The CNN architectures that we plot (left) are DenseNet121 (Huang et al., 2017), InceptionV3 (Szegedy et al., 2016), ResNet50 (He et al., 2016), and ResNeXT50 (Xie et al., 2017). On the right we plot hidden layers from transformers: ViT (Dosovitskiy et al., 2020) and Swin (Liu et al., 2021). All use the default ImageNet torchvision (Marcel and Rodriguez, 2010) weights. We note two trends in these plots:

1. All curves consist of a plateau spanning most layers of the model followed by a dramatic dropoff in stable rank at the last layers.
2. The transfomer models exhibit significantly more *fluctuation* in stable rank than the CNNs.

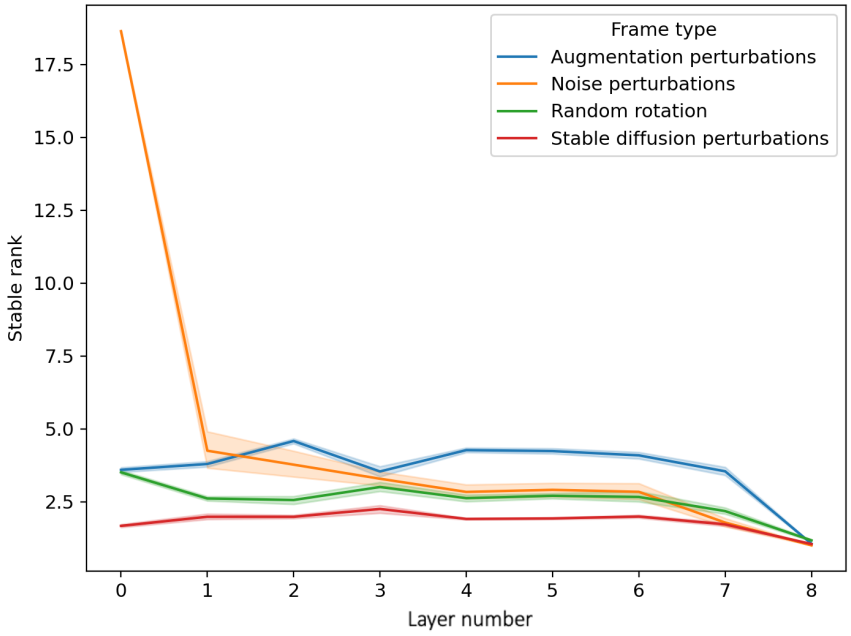

Figure 8: The stable rank of different types of frames measured at various layers of a ViT model. The ResNet50 version of this plot is found in Figure 2. Layer zero corresponds to model input and the last layer corresponds to model output. Shaded regions indicate 95% confidence intervals over 40 randomly selected ImageNet images.

1 could be partially explained by the fact that *all* models studied in this paper, both CNNs and transformers, include residual connections. Note that a toy residual network with n-dimensional feature spaces and identity activations consists of a composition of layers of the form $I_n + W$, where I_n is an identity matrix and W a $n \times n$ weight matrix. These have singular values of the form $1 + \sigma_i$, where $\{\sigma_i\}$ are the singular values of W, and thus stable rank

$$\frac{1}{(1+\sigma_1)^2} \sum_i (1+\sigma_i)^2. \tag{2}$$

Suppose W is a random matrix with IID entries sampled from $\mathcal{N}(0, \frac{2}{n})$ (this is true before training with He normal initialization (He et al., 2016)). Then a calculation using (Marčenko and Pastur, 1967) shows that for large n the expected stable rank of $I_n + W$ is approximately

$$\frac{n}{(1+2\sqrt{2})^2} \int_0^{2\sqrt{2}} (1+y^2)\sqrt{8-y^2}\frac{dy}{2\pi} \approx 0.37n. \tag{3}$$

Provided this is larger than the stable rank of the neural frame in the preceding layer, Lemma 7 might suggest that $I_n + W$ preserves the stable rank of the neural frame.[2]

As for 2, while these experiments alone are insufficient to identify a reason for this apparent difference, one could make a number of different conjectures. It could be, for instance, that the priors hardcoded into CNNs dampen the extent to which the models stretch and compress an image manifold. Alternatively, it might be that the nonlinearity of attention layers leads to more geometric changes layer-to-layer. This nonlinearity is in contrast to the convolutions in CNNs, which are linear in isolation (that is, not considering the nonlinear layers that often follow them). These questions would be interesting to address in follow-up work.

Appendix H. Geometric Background

In this paper we take the word 'manifold' to mean a smooth manifold in the formal sense. A comprehensive reference on manifolds is (Lee, 2013) — here for convenience we briefly introduce key geometric objects of interest: tangent bundles, their sub-bundles, and frames.

Let $M \subset \mathbb{R}^n$ be an m-dimensional smooth manifold, x a point on M, and suppose $\gamma : (-1, 1) \to M$ is a smooth path on M such that $\gamma(0) = x$. The *tangent vector* associated with γ at x is the derivative of γ at 0, $\gamma'(0)$. The tangent space T_xM of M at x is the vector space of all such tangent vectors $\gamma'(0)$ at x (here γ varies over all possible smooth paths in M that pass through x). The *tangent bundle* TM of M is the union of all tangent spaces for each $x \in M$, $\coprod_{x \in M} T_xM$ — it is a manifold in its own right of dimension $2m$. For more details see Section 3 in (Lee, 2013).

The tangent bundle can be thought of as the assignment of a vector space to each point in M; the concept of a vector bundle generalizes the tangent bundle of a manifold. Informally, a vector bundle is a smooth map $\pi : E \to M$ such that the fibers $E_x := \pi^{-1}(x) \subseteq E$ are finite-dimensional real vector spaces isomorphic to $\mathbb{R}^l$ and they vary smoothly with respect to x in the sense that for any $x \in M$ there is a neighborhood U of x with a diffeomorphism $\varphi_x : \pi^{-1}(U) \to U \times \mathbb{R}^l$. A sub-vector bundle $F \subseteq E$ is an embedded submanifold which is itself a vector bundle over M (with respect to the induced smooth map $F \subseteq E \xrightarrow{\pi} M$) such that for each point $x \in M$, $F_x \subseteq E_x$ is a linear subspace. For formal definitions we refer to Section 10 in (Lee, 2013). Our interest is in sub-vector bundles of tangent bundles TM.

A common way to obtain such sub-vector bundles in practice is from vector fields. Recall that in the case of the tangent bundle, the map $\pi : TM \to M$ is the map such that if $z \in T_xM \subset TM$, then $\pi(z) = x$. Then a smooth vector field is formally a smooth function $v : M \to TM$ with the property that $\pi(v(x)) = x$ for all $x \in M$ (informally this corresponds to the usual notion that a vector field consists of a choice of tangent vector for each $x \in M$). We can use parametrized functions from our manifold to itself to construct vector fields.

Lemma 4 (Prop 9.7 (Lee, 2013)) *If $f : (-1, 1) \times M \to M$ is a smooth function on smooth manifold M with the property that $f(0, x) = x$ for all $x \in M$, then the function $v(x) = \frac{\partial}{\partial t} f(t, x)|_{t=0}$ is a smooth vector field.*

2. The expected stable rank of W itself is $\frac{n}{4}$.

Lemma 5 *Let $v_1, \ldots, v_k : M \to TM$ be smooth vector fields on an m-dimensional manifold M.*

1. *The set U of all x in M such that $v_1(x), \ldots, v_k(x)$ are linearly independent is open.*[3]
2. *The (sub)spaces $\operatorname{span}(v_1(x), \ldots, v_k(x)) \subseteq T_xM$ form a sub-vector bundle of the tangent bundle of U.*

We put the two lemmas above together to give the statement that will form the basis for one type of neural frames that we introduce in the next section.

Corollary 6 *Suppose that $\mathcal{F} = \{f_i : (-1,1) \times M \to M \,|\, i = 1, \ldots, k\}$ is a collection of smooth maps such that $f_i(0,x) = x$ for all $x \in M$, and let $v_i(x) = \frac{\partial}{\partial t} f_i(t,x)|_{t=0}$. If $v_1(x), \ldots, v_k(x)$ are linearly independent in T_xM for all $x \in M$, then $f_1, \ldots, f_k$ define a k-dimensional vector bundle on M which we denote by $V_{\mathcal{F}}$ and $v_1(x), \ldots, v_k(x)$ is a k-frame of this vector bundle.*

The geometric machinery we have introduced in this section will provide the framework for neural frames, which we introduce below. We note however that this framework is supposed to act as a guide, not a guarantee. Indeed, by necessity, we will have to violate certain assumptions when running experiments. For example, many popular deep learning architectures are not actually smooth everywhere and some of our augmentations will not be smooth either (largely due to the discrete nature of digital images). However, we have tried to choose functions that are at least moderately well-behaved.

H.1. Stable rank and multiplication of matrices

It is worth mentioning that while true linear algebraic rank has the property that the rank of a product AB is at most the minimum of the ranks of A and B, this fails for stable rank. However, the extent of this failure is controlled by the behavior of the top singular values of A, B and AB — precisely:

Lemma 7 *If A and B are $l \times m$ and $m \times n$ matrices respectively, then*

$$r(AB) \leq \Big(\frac{\|A\|_{\text{spec}} \|B\|_{\text{spec}}}{\|AB\|_{\text{spec}}} \Big)^2 \min\{r(A), r(B)\}. \tag{4}$$

We include Lemma 7 since when taking $A = dF_{i+1}$ and $B = dF_{\leq i}$, so that $AB = dF_{\leq i+1}$, Lemma 7 seems somewhat explanatory of the general decreasing trend in the curves of, for example, Figure 2 (right). Note that by the multiplicative property of the spectral norm, the first factor on the right hand side of equation 4 is always ≥ 1.

Appendix I. Proofs

The statement of Lemma 4 and the proof below is very similar to those of Proposition 9.7 in (Lee, 2013), however note that our hypotheses on f are weaker.

3. although possibly empty.

Proof [Proof of Lemma 4] First, by Proposition 3.14 in (Lee, 2013) there is a natural decomposition of tangent bundles

$$T\big((-1,1)\times M\big) \simeq T(-1,1)\times TM \tag{5}$$

and so by Proposition 3.21 in (Lee, 2013) the differential of f can be naturally identified as a smooth map

$$df : T(-1,1)\times TM \to TM. \tag{6}$$

Now as $T(-1,1)\simeq(-1,1)\times\mathbb{R}$ (for example combining Proposition 3.9 and 3.13 in (Lee, 2013)), letting $z : M \to TM$ denote the zero section we can define a smooth map $\frac{\partial f(t,x)}{\partial t}|_{t=0}$ as

$$\begin{aligned} &M \xrightarrow{\sigma} (-1,1)\times\mathbb{R}\times TM \simeq T(-1,1)\times TM \xrightarrow{df} TM \\ &\text{sending } x \mapsto (0,1,z(x)) \mapsto df((0,1),z(x)) \end{aligned} \tag{7}$$

(one can check in coordinates that this is indeed a partial derivative with respect to t, hence the notation). It remains to check that if $\pi : TM \to M$ is the projection, we have $\pi(\frac{\partial f(t,x)}{\partial t}|_{t=0}) = x$, and this follows from the hypothesis that $f(0,x) = x$ for all $x \in M$. ■

Proof [Proof of Lemma 5] The statement is local on M. We may therefor assume that $M \subseteq \mathbb{R}^m$ is an open subset of some euclidean space, in which case we have a canonical identification $TM \simeq M\times\mathbb{R}^m$. Since a section of the projection $\pi : M\times\mathbb{R}^m \to M$ is equivalent to a function $M\to\mathbb{R}^m$, we may now identify $v_1,\dots,v_k$ as smooth functions

$$\begin{aligned} &v_1,\dots,v_k : M\to\mathbb{R}^m, \text{ whose product is a smooth map} \\ &(v_1,\dots,v_k) : M \to (\mathbb{R}^m)^k \end{aligned} \tag{8}$$

Finally, let $m_J : (\mathbb{R}^m)^k \to \mathbb{R}$ be the $k\times k$ minors of $m\times k$ matrices, where $J\subset\{1,\dots,m\}$ ranges over subsets of cardinality k — these are polynomials and hence smooth. The set

$$\begin{aligned} &\{(w_1,\dots,w_k)\in(\mathbb{R}^m)^k \\ &\mid w_1,\dots,w_k \text{ are linearly independent}\} \end{aligned} \tag{9}$$

can be identified as the locus where $\prod_J m_J(w_1,\dots,w_k)\neq 0$; since the non-vanishing locus of a smooth function is open, we have proved part (i) of the lemma.

For part (ii), by the very definition of U the functions $v_1,\dots,v_k$ define a global isomorphism

$$\begin{aligned} &\varphi : U\times\mathbb{R}^k \to \operatorname{span}(v_1,\dots,v_k) \\ &\text{where } \varphi(x,(c_1,\dots,c_k)) = \sum_i c_i v_i(x). \end{aligned} \tag{10}$$

Using this, it is straightforward to check that $\operatorname{span}(v_1,\dots,v_k)$ is a vector bundle over U. ■

Remark 8 *All of the above proofs work even if the v_i are continuous (smoothness is not required). The proof of (ii) in fact shows the stronger statement that* $\mathrm{span}(v_1, \ldots, v_k)$ *is a* trivial *vector bundle over U. It is worth mentioning that in the* global *context where Lemma 5 is stated, there are examples where it is impossible that $U = M$, even when $k = 1$: for example, the famous "hairy ball theorem" says that when $M = S^2$ (a 2-sphere), every globally defined vector field $v : M \to TM$ vanishes at at least one point.*

We sketch here a statement making precise the sense that for almost all sets of $k \leq \dim M$ vector fields $v_1, \ldots, v_k : M \to TM$ the locus

$$\{x \in M \,|\, v_1(x), \ldots, v_k(x) \text{ are linearly independent}\} \tag{11}$$

is a dense open set with measure 0 complement. Let $\Gamma(M, TM)$ be the real vector space of vector fields on M. We will make use of the *relative* k-fold product $\prod_{M,i=1}^{k} TM$ of TM over M; this is simply the space of k-tuples of tangent vectors $w_1, \ldots, w_k \in TM$ such that $\pi(w_i) = \pi(w_j) \in M$ for all i, j. The relative product contains a subspace $N \subseteq \prod_{M,i=1}^{k} TM$ consisting of linearly dependent k-tuples $(w_1, \ldots, w_k)$. Note that this subspace is *not* a submanifold: on each fiber $\prod_{i=1}^{k} T_xM$ it is a *vanishing locus* of a product $\prod_J m_J(w_1, \ldots, w_k)$ of minors as appearing in the above proof.[4] However, it does admit a *stratification*

$$N_0 \subseteq N_1 \subseteq \ldots, \subseteq N_{k-1} = N \tag{12}$$

such that $N_i \setminus N_{i-1}$ is a (not necessarily closed) submanifold of $\prod_{M,i=1}^{k} TM$ for $i = 1, \ldots, k-1$. Namely, one defines N_i to consist of the $(w_1, \ldots, w_k)$ whose associated $m \times k$ matrix has rank less than or equal to i.

Claim: Let $V \subset \Gamma(M, TM)$ be a finite dimensional linear subspace, and assume that for each $x \in M$ the natural linear map $\Psi : V \to T_xM$ defined as $\Psi(v) = v(x)$ is surjective. Then, for almost every $(v_1, \ldots, v_k) \in V^k$ the set defined in equation 11 is a dense open set with measure 0 complement.

Note that in Claim I, the fact that Ψ is surjective for each $x \in M$ does not imply that V is equal to $\Gamma(M, TM)$ (for example, $\Gamma(M, TM)$ will generally be infinite dimensional). Rather, for any $v \in T_xM$ there is a vector field that takes value v at x. We will not provide a full proof of this claim, merely a sketch. To begin, consider the natural map

$$\Phi : V^k \times M \to \prod_{M,i=1}^{k} TM \tag{13}$$

defined by $\Phi((v_1, \ldots, v_k), x) \mapsto (v_1(x), \ldots, v_k(x))$. *If N were* a smooth closed submanifold of $\prod_{M,i=1}^{k} TM$ (as stated above, it isn't, this is just a thought experiment to motivate a proof sketch), then we could proceed as follows: the condition on Ψ could be used to show that the map Φ is transverse to N. Then, the parametric transversality theorem (Theorem 6.35 in (Lee, 2013)) would imply that for almost all $(v_1, \ldots, v_k) \in V^k$, the resulting map

$$\sigma : M \to \prod_{M,i=1}^{k} TM \tag{14}$$

4. That is, one can verify that N is singular using the Jacobian criterion.

defined by $\sigma(x) = \Phi((v_1, \ldots, v_k), x)$ is transverse to N. Noting that N has codimension 1, we would conclude that its preimage $\sigma^{-1}(N) \subseteq M$ is a closed submanifold of codimension 1 (hence a set of measure 0). Since the set defined in 11 is $M \setminus \sigma^{-1}(N)$ this would complete the proof.

Again, this proof sketch is merely a heuristic as we know that N is not smooth. However, there exist *stratified* transversality theorems (hence the discussion of a stratification of N), and replacing our heuristic application of the more well known parametric transversality theorem with one of these (for example the main theorem of (Trotman, 1978/79), see also references therein) yields a proof strategy.

Proof [Proof of Lemma 7] We first reduce to the case where all matrices are square: if not, letting $p = \max\{l, m, n\}$ we may embed A and B in the upper left block of $p \times p$ matrices, and direct calculation shows

$$\begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} AB & 0 \\ 0 & 0 \end{pmatrix}. \tag{15}$$

Moreover, such padding by zeros does not alter singular values, hence leaves all stable ranks in sight unchanged.

From now on we assume A, B and thus AB are $n \times n$ matrices. Let $s(A) = (s(A)_1, \ldots, s(A)_n)$ be the n-dimensional vector of sorted singular values of A (and similarly for B and AB). By Theorem IV.2.5 in (Bhatia, 1996),

$$s(AB)^2 <_w s(A)^2 s(B)^2 \tag{16}$$

where $<_w$ denotes weak submajorization, and the product on the right hand side is coordinatewise (a.k.a. Hadamard) multiplication. From this and the definition of weak submajorization in Section II.1 (Bhatia, 1996)) it follows that

$$\begin{aligned} &\sum_{i=1}^{n} s(AB)_i^2 \leq \sum_{i=1}^{n} s(A)_i^2 s(B)_i^2 \leq s(A)_1^2 \sum_{i=1}^{n} s(B)_i^2 \\ &= (s(A)_1 s(B)_1)^2 \frac{1}{s(B)_1^2} \sum_{i=1}^{n} s(B)_i^2 \end{aligned} \tag{17}$$

where the first inequality follows from the definition of weak submajorization in Section II.1 (Bhatia, 1996)) and the second inequality uses the fact that $s(A)_1 \geq \cdots \geq s(A)_n$.[5] We now divide both sides of 17 by $s(AB)_1^2$ to obtain

$$\begin{aligned} r(AB) &= \frac{1}{s(AB)_1^2} \sum_i s(AB)_i^2 \\ &\leq \frac{(s(A)_1 s(B)_1)^2}{s(AB)_1^2} \frac{1}{s(B)_1^2} \sum_i s(B)_i^2 \\ &= \Big(\frac{\|A\|_{\text{spec}} \|B\|_{\text{spec}}}{\|AB\|_{\text{spec}}} \Big)^2 r(B). \end{aligned} \tag{18}$$

5. Alternatively, this is the $p = \infty$, $q = 1$ Hölder's inequality.

By a symmetric argument,[6]

$$r(AB) \leq \left(\frac{\|A\|_{\text{spec}}\|B\|_{\text{spec}}}{\|AB\|_{\text{spec}}}\right)^2 r(A). \tag{19}$$

■

Proof [Proof of Proposition 3] This follows directly from Theorem 3.26 in (Kirillov, 2008), which shows that the Lie algebra of G_x is the kernel of the natural map $\mathfrak{g} \to T_x M$. Hence if this natural map is injective, G_x has 0-dimensional Lie algebra and is thus a 0-dimensional closed subgroup of G, i.e. a discrete subgroup. ■

Appendix J. Experimental details

We ran all of our experiments on an Nvidia A100 GPU. The specific hyperparameters that we used in training can be found in Table 13.

J.1. Augmentation frames

In Table 1 we describe the augmentations that we use in the experiments with augmentation frames in this paper, the software library used to implement them, and the parameter settings that we used to approximate the tangent vector corresponding to each.

J.2. Handling edge effects in image translation and rotation

Because images have finite support, pixels with zero value appear at the edges when the image is rotated by θ degrees (where $\theta \neq 0°, 90°, 180°$, or $270°$). Thus, these augmentations violate our goal of only using augmentations that produce naturalistic images. To handle this situation in practice, we increase the size of the image so that we can crop out fractional numbers of pixels (after rotation). In detail we:

1. resize the image to $\times 8$ its original size,
2. rotate the image by 5 degrees,
3. crop out empty pixels at the corners by removing 20 pixels around the border,
4. resize the image back to its original size.

J.3. Hidden Layers Used for Each Architecture

A significant amount of this work depends on querying the hidden representations of different models. In Tables 2-10 we list the hidden layers used in each model type in the experiments.

Table 1: Augmentation transformations and the library and parameters used when implementing them.

Augmentation	Library	Default parameters
JPEG transform	imgaug library (Jung et al., 2020)	Compression 70%
Brightness	Torchvision	Brightness factor 1.02
Crop and resize	Torchvision	Resize $\times 4$, crop off 1 pixel, Interpolation: bilinear, nearest, linear
Contrast	Torchvison	Contrast factor 1.05
Gamma transform	Torchvision	Gamma 1.02
Hue	Torchvision	Hue factor .01
Saturation	Torchvision	Saturation factor 1.1
Sharpness	Torchvision	Sharpness factor 1.2
Downscale	Torchvsion	Resize $\times 0.9$, return to original Interpolation: bilinear, nearest, linear
Rotation + translation	Torchvision	Resize $\times 4$, rotate 2 degrees, centers $(0,0),(50,50),(-50,50)$
Gaussian blur	Torchvision	Kernel size 3×3, $\sigma = 2.0$
Log correction	Kornia (**?**)	Gain 1.05
Sigmoid transform	Kornia	Cutoff 0.5, gain 5

Table 2: Layers for torchvision ViT Base 16.

Layer name	Layer number
`encoder.layers.encoder_layer_1.mlp`	1
`encoder.layers.encoder_layer_3.mlp`	2
`encoder.layers.encoder_layer_5.mlp`	3
`encoder.layers.encoder_layer_7.mlp`	4
`encoder.layers.encoder_layer_9.mlp`	5
`encoder.layers.encoder_layer_11.mlp`	6
`getitem_5`	7

Appendix K. What Can Frame CKA Tell Us?

In Section 3 we noted that frame CKA can illuminate some small-scale properties of a model's representations that are hard to detect with standard CKA. In Figure 16 we show that frame CKA detects the difference in representations between models with adversarial

6. For example, use the facts that $(AB)^T = B^T A^T$ and transposing doesn't change singular values.

Table 3: Layers for torchvision Swin T.

Layer name	Layer number
features.1.0.mlp	1
features.3.0.mlp	2
features.5.0.mlp	3
features.5.2.mlp	4
features.5.4.mlp	5
features.7.0.mlp	6
features.7.1.mlp	7
flatten	8

Table 4: Layers for torchvision ResNeXT50 32×4d.

Layer name	Layer number
layer1.0.add	1
layer2.0.add	2
layer3.0.add	3
layer4.0.add	4
flatten	5

training and models without. Namely, neural frame CKA exhibits that inter-architecture similarity increases with adversarial robustness, in the sense that similarity between layers of different models increases as the ϵ used in adversarial training increases. For example, the similarity between layers is larger between $\epsilon = 3$ and $\epsilon = 5$ than between $\epsilon = 0.1$ and $\epsilon = 5$. This was only previously shown with an expensive deconfounding variant of CKA in (Jones et al., 2022).

In another example, we compare the intra-layer similarity between a ResNet50 and a Vision Transformer. Prior work with CKA had noted that CKA tends to show significant differences between blocks in ResNets but less intra-layer differences in vision transformers. In Figure 16 we show that frame CKA (using augmentation frames) picks up a different signal than standard CKA. We see that frame CKA actually shows greater differences between layers in the ViT suggesting that the outcome of a comparison of these two model types may depend on the scale at which one compares their representations. We note that our observations are consistent with findings in Section G.

Table 5: Layers for torchvision MobileNetV3 (small).

Layer name	Layer number
features.1.block.2	1
features.3.add	2
features.5.add	3
features.7.block.3	4
features.9.block.3	5
features.11.add	6
flatten	7
classifier.0	8
classifier.1	9
classifier.2	10

Table 6: Layers for torchvision InceptionV3.

Layer name	Layer number
Conv2d_1a_3x3.relu	1
Conv2d_2b_3x3.relu	2
Conv2d_4a_3x3.relu	3
Mixed_5b.cat	4
Mixed_5d.cat	5
Mixed_6a.cat	6
Mixed_6c.cat	7
Mixed_6e.cat	8
Mixed_7a.cat	9
Mixed_7b.cat_2	10
flatten	11

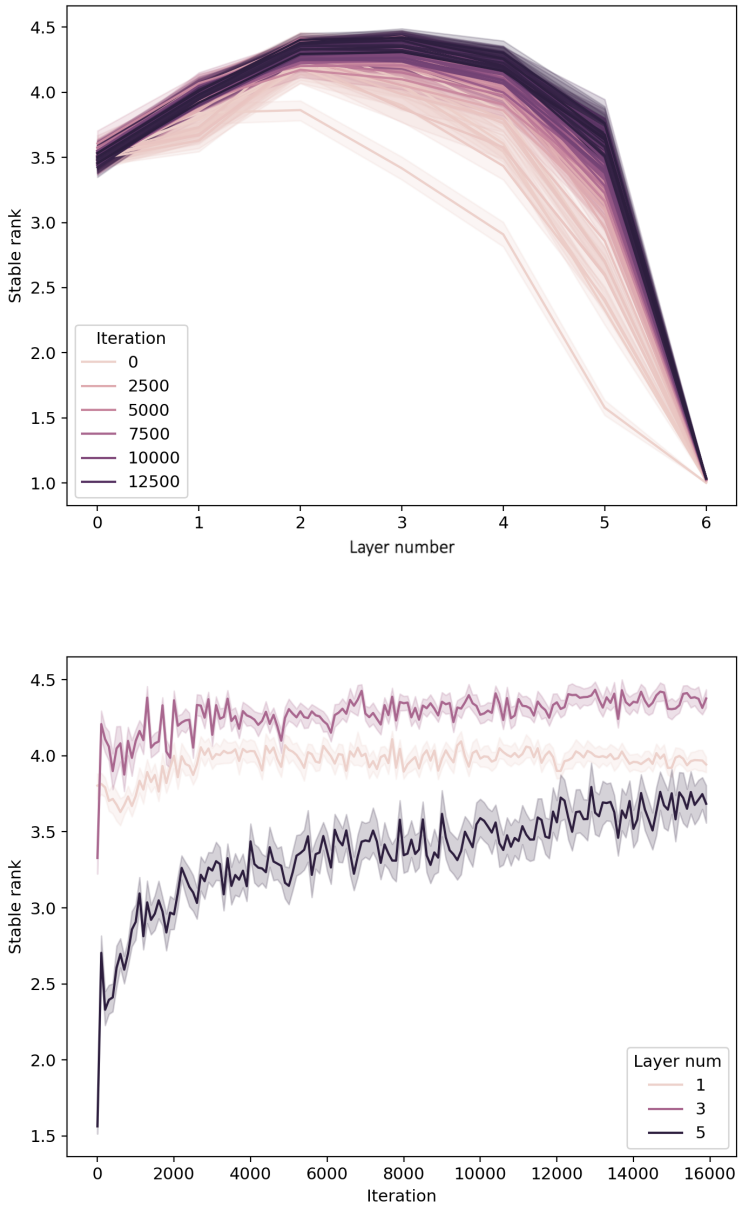

Figure 9: **(Left)** The stable rank of an augmentation frame (by layer) for a ResNet18 trained from scratch. Different colored curves correspond to the number of iterations of training that the model has undergone. **(Right)** The stable rank of different layers of the model as a function of the number of training iterations. Shaded regions in both plots indicate 95% confidence intervals over 40 random ImageNet images.

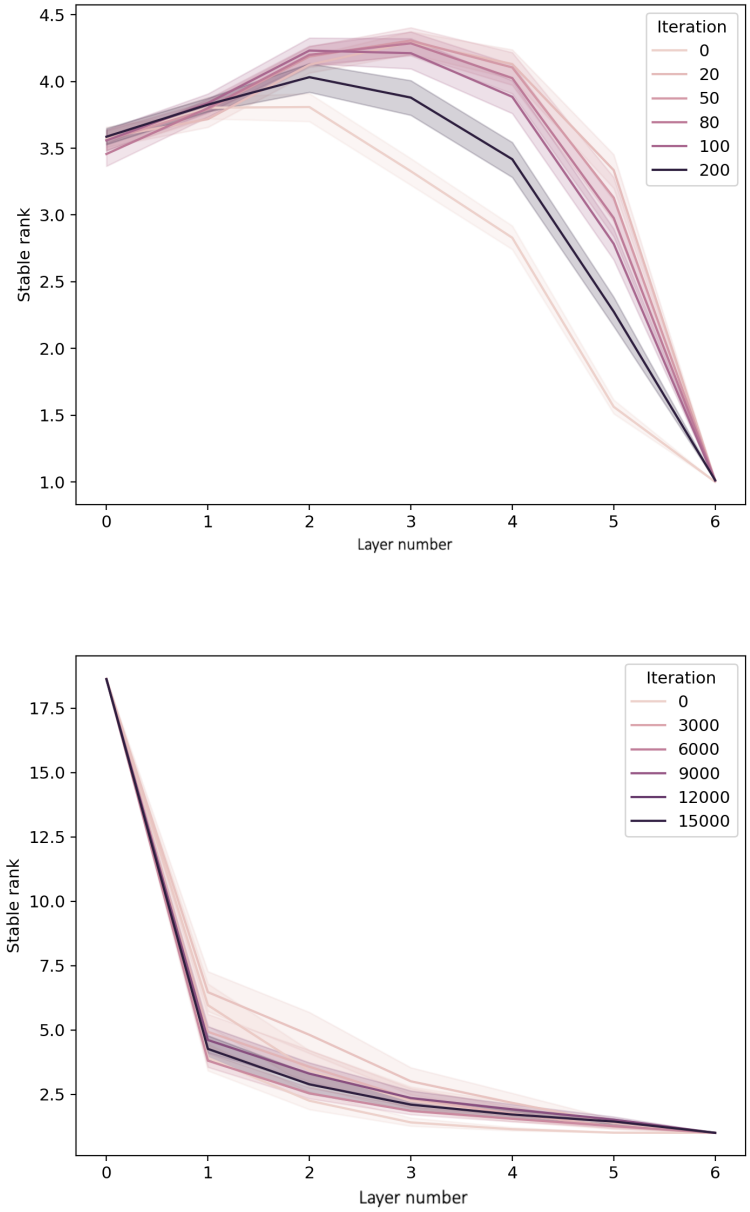

Figure 10: **(Left)** The stable rank of an augmentation frame (by layer) during early iterations of a ResNet18 trained from scratch. **(Right)** The stable rank of the ResNet18 model (by layer) evaluated on a noise frame. In both plots, different colored curves correspond to the number of iterations of training that the model has undergone.

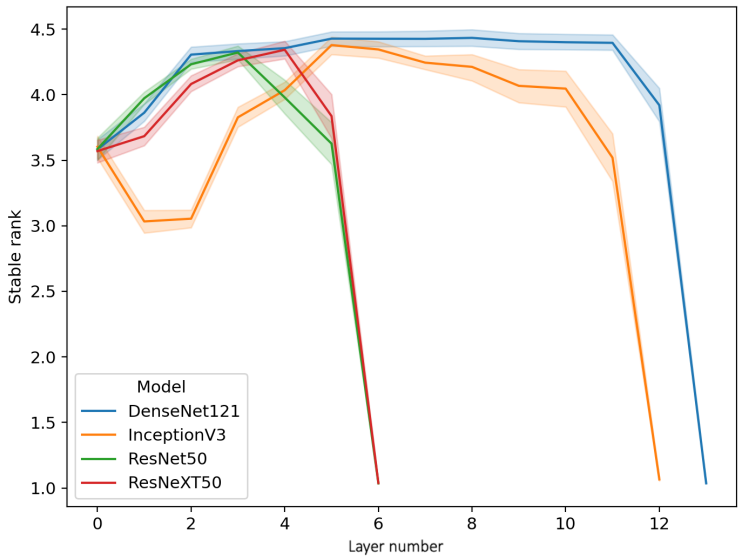

Figure 11: The stable rank, by layer, for a range of different CNN architectures. Shaded regions indicate 95% confidence intervals over 40 random ImageNet images.

Table 7: Layers for torchvision DenseNet121.

Layer name	**Layer number**
`features.denseblock1.denselayer2.cat`	1
`features.denseblock2.denselayer1.cat`	2
`features.denseblock2.denselayer7.cat`	3
`features.denseblock2.cat`	4
`features.denseblock3.denselayer6.cat`	5
`features.denseblock3.denselayer12.cat`	6
`features.denseblock3.denselayer18.cat`	7
`features.denseblock3.cat`	8
`features.denseblock4.denselayer6.cat`	9
`features.denseblock4.denselayer12.cat`	10
`features.denseblock4.cat`	11
`flatten`	12

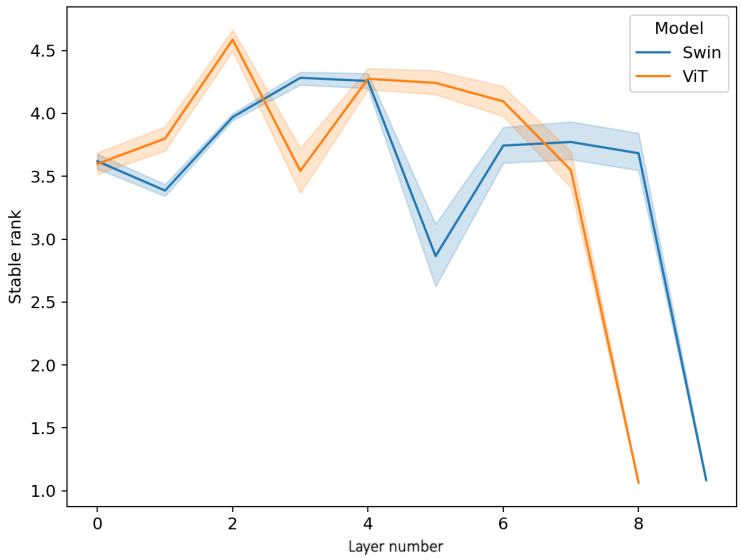

Figure 12: The stable rank, by layer, for a range of different transformer architectures. Shaded regions indicate 95% confidence intervals over 40 random ImageNet images.

Table 8: Layers for timm resnetv2 101×1 (BiT).

Layer name	**Layer number**
`stages.0.blocks.0.add`	1
`stages.1.blocks.0.add`	2
`stages.2.blocks.0.add`	3
`stages.2.blocks.4.add`	4
`stages.2.blocks.8.add`	5
`stages.2.blocks.12.add`	6
`stages.2.blocks.16.add`	7
`stages.2.blocks.20.add`	8
`stages.3.blocks.2.add`	9
`head.global_pool.flatten`	10

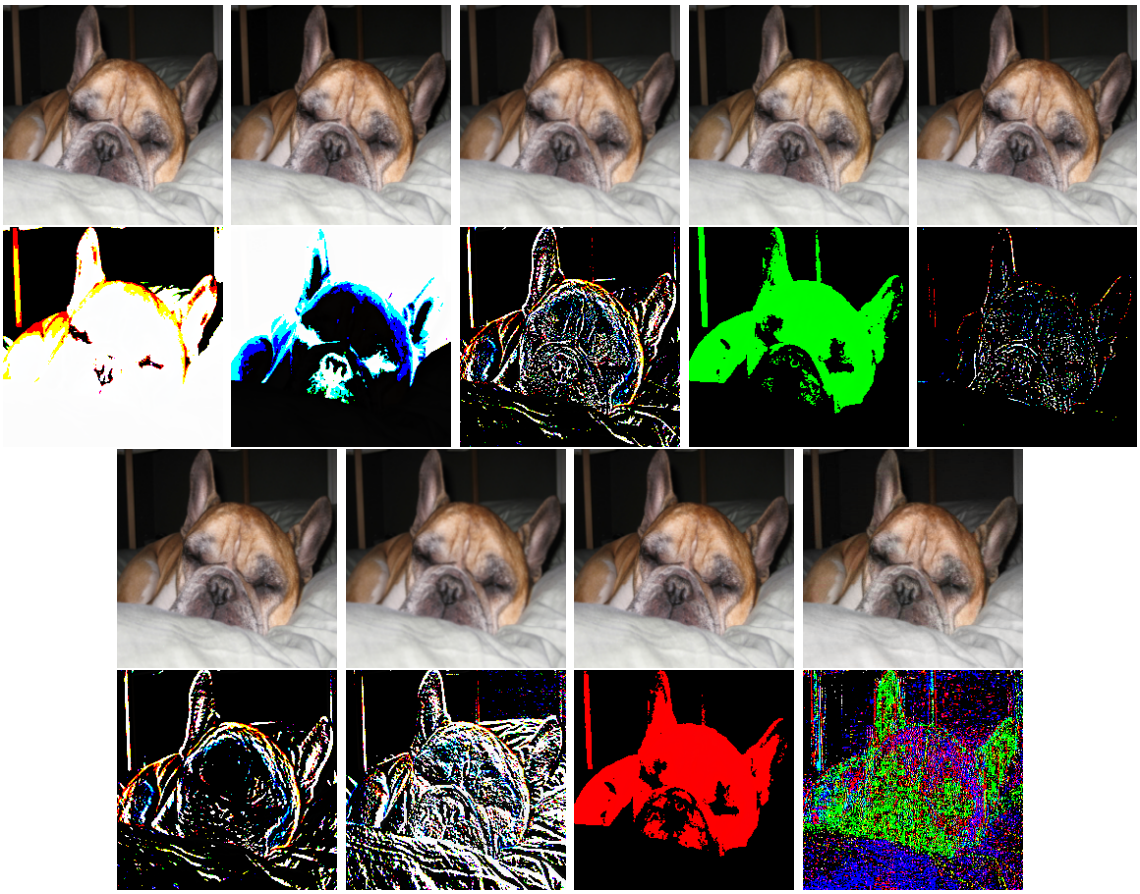

Figure 13: **(First and third rows)** A subset of the sample perturbations, from left to right and top to bottom: brightness, contrast, crop with bilinear interpolation, hue, sharpness, crop with linear interpolation, rotation, saturation, jpeg compression. **(Second and fourth rows)** The difference between the original and perturbed images (above).

Table 9: Layers for torchvision ConvNeXT small.

Layer name	Layer number
`features.1.0.block.0`	1
`features.3.0.block.0`	2
`features.5.0.block.0`	3
`features.5.8.block.0`	4
`features.5.16.block.0`	5
`features.7.1.block.0`	6
`classifier.1`	7

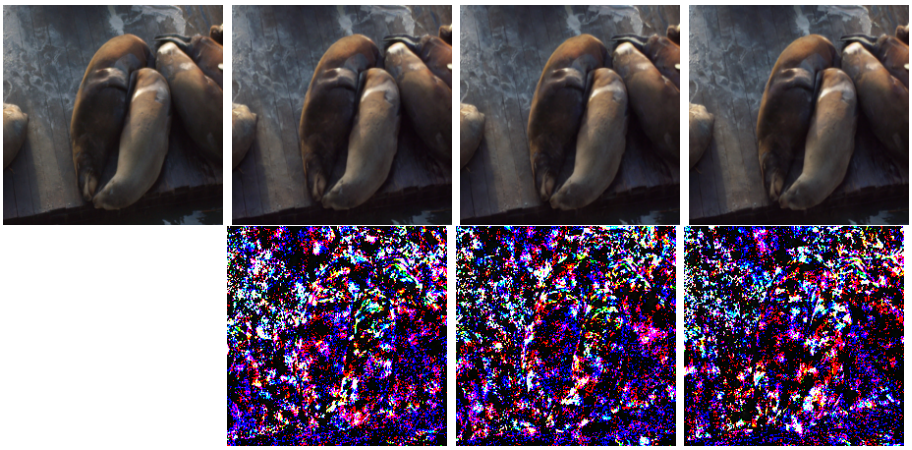

Figure 14: **(First row)** The original ImageNet image and perturbations of this image sampled using the Boomerang method and stable diffusion. **(Second row)** The difference between the original image and each perturbation.

Table 10: Layers for torchvision AlexNet.

Layer name	**Layer number**
`features.1`	1
`features.4`	2
`features.7`	3
`features.9`	4
`features.11`	5
`classifier.2`	6
`classifier.5`	7
`flatten`	8

Table 11: Layers for torchvision ResNet50.

Layer name	**Layer number**
`layer1.2.relu_2`	1
`layer2.3.relu_2`	2
`layer3.5.relu_2`	3
`layer4.2.relu_2`	4
`flatten`	5

Table 12: Layers for torchvision ResNet18.

Layer name	**Layer number**
layer1.1.relu_1	1
layer2.1.relu_1	2
layer3.1.relu_1	3
layer4.0.relu_1	4
flatten	5

Table 13: Hyperparameters used when training the ResNet18 on ImageNet.

Training hyperparameters	
Optimzer	SGD
Learning rate	0.5
Learning rate scheduler	cyclic
Learning rate peak epoch	2
Momentum	0.9
Batch size	1024
Epochs	16
Weight decay	5e−5
Label smoothing	0.1
BlurPool?	Yes
Pretrained?	No

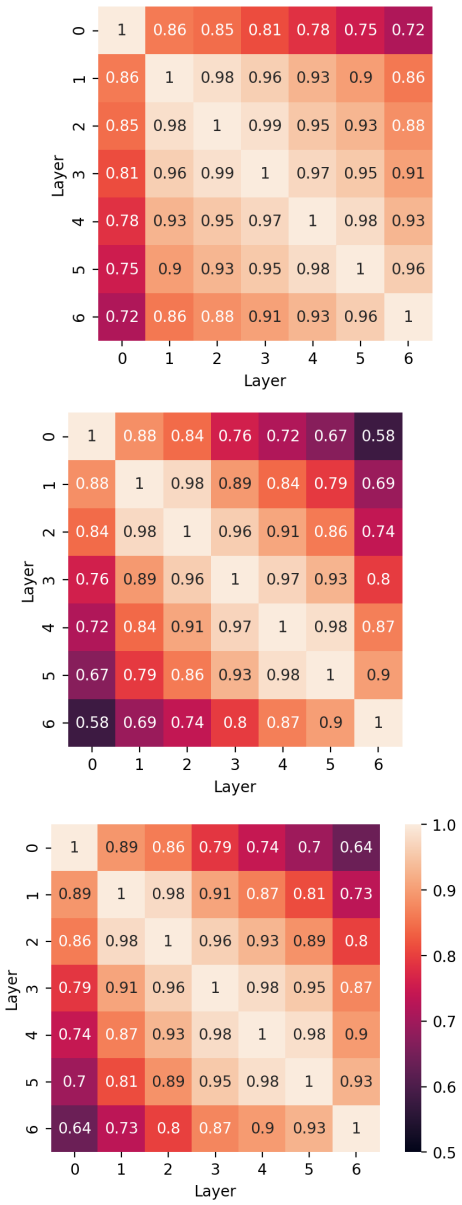

Figure 15: The frame CKA scores between layers in the same model when augmentation frames are used. (**Top**) an ImageNet trained vanilla ResNet50 (Marcel and Rodriguez, 2010), (**Middle**) an adversarially trained ImageNet ResNet50 with $\epsilon = 5$, (**Bottom**) an ImageNet trained ResNet50 with Deep Augmentation. One can see that both heavy augmentation and adversarial training cause a model's representations to increasingly vary between layer. This effect appears to be stronger for adversarial training than heavy augmentation.

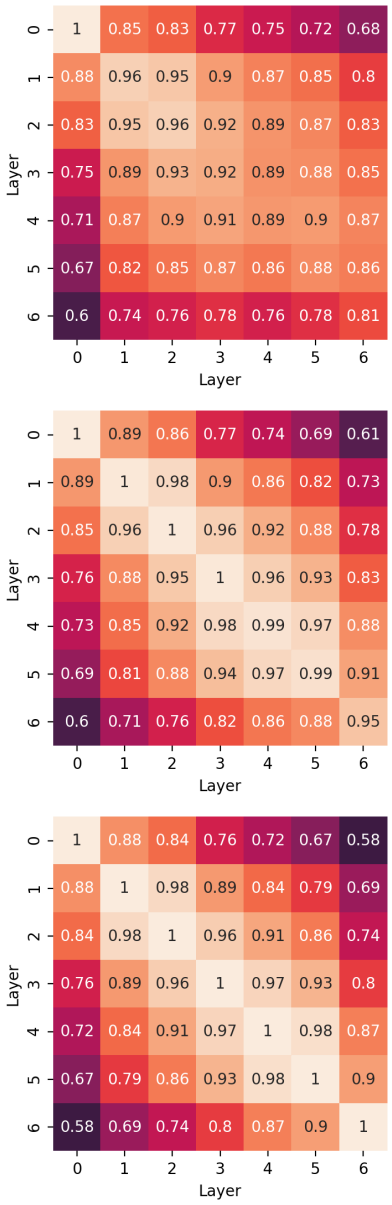

Figure 16: Heatmaps for frame CKA with augmentation frames for three different models (all trained on ImageNet) compared to a ResNet-50 trained on ImageNet with $\epsilon = 5$ adversarial training: (**Top**) a ResNet50 trained with $\epsilon = 0.1$ adversarial training, (**Middle**) a ResNet50 trained with $\epsilon = 3$ adversarial training, (**Bottom**) and the same ResNet50 trained with $\epsilon = 5$ adversarial training.

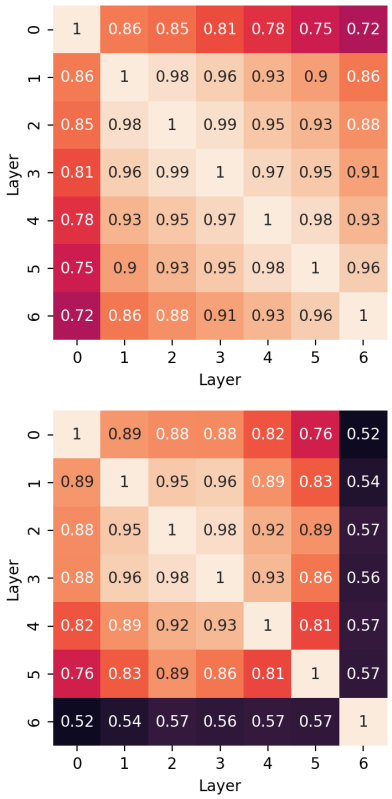

Figure 17: Heatmaps for frame CKA with augmentation frames for ResNet50 vs. ResNet50 trained on ImageNet (**Top**) and ViT vs ViT (Dosovitskiy et al., 2020) trained on ImageNet (**Bottom**).