On the hidden layer-to-layer topology of the representations of reality realised within neural networks

Oliver Gafvert[1], Peter Grindrod CBE[1], Heather A. Harrington[1], Catherine F. Higham[2], Desmond J. Higham[3], and Ka Man Yim[4]

[1]Mathematical Institute, University of Oxford, Oxford, United Kingdom
[2]School of Computing Science, University of Glasgow, Glasgow, United Kingdom.
[3]School of Mathematics, University of Edinburgh, Edinburgh, United Kingdom.
[4]School of Mathematics, University of Cardiff, Cardiff, United Kingdom

December 9, 2024

Abstract

Consider an information processing algorithm that is designed to process an input data object onto an output data object via a number of successive internal *layers*, and mappings between them. The possible activation state within each layer can be represented as a cube within Euclidean space of a high dimension (e.g. equal to the number of artificial neurons at that level). Multiple instances of such input objects produce a point cloud within each layer's cube: this is the "representation of the reality" at that layer, as sampled by the set of input objects.

Most neural networks reduce the dimension of each layer's cube from layer to successive layer. This gives the false impression of refining the inner representations of reality, distilling it down to fewer dimensions from which to discriminate or to infer outcomes (whatever is the aim). However the representation of reality realised within each layer's cube is a manifold, a curved subset embedded within it and of much lower dimension. Investigations show that such manifolds may not always be reducing in their local dimension. Instead the manifold may become folded over and over, filling up further dimensions, and creating non-realistic (unforeseeable) proximities. This type of feature is likely to be generic (universal).

We discuss some of the likely consequences of these relatively unforeseen characteristics, and, in particular, the possible vulnerability of such algorithms to non-realistic perturbations. We discuss an appropriate response.

Keywords: Deep learning networks, topological data analysis, persistent homology

1 Introduction

Consider an information processing algorithm, such as a deep neural network, that is designed to process an input data object onto an output data object, usually making a discrete classification, via a number of successive internal *layers*, and mappings between them. Such algorithms are high-dimensional, often with millions of real parameters to be calibrated during training; they are non-linear, and the overall mapping from input space to output space may have some high gradients [1].

Whenever such an algorithm is applied to an input data object, each successive kth layer produces a set of observable, internal, real values that are usually called the "activations" of the layer's set of artificial neurons. These may be realised as a vector within a corresponding Euclidean space of high dimension, m_k say. We will refer to this space as the *embedding space*, E_k, and to m_k as the *embedding dimension*, which is also often called the *width* of the kth layer. The dimensions m_k of each E_k will be known. When the algorithm is applied to a large number, N, of input data objects it yields N vectors forming a point cloud within E_k. As we increase N we can think of the point cloud within E_k as more and more densely sampling a manifold $M_k \subset E_k$. Typically M_k will be curved and have a dimension that is much lower than m_k: it

is the image of the input object space held at the kth layer. Of course there is also a (latent) continuous mapping, ϕ_k say, that maps the M_k onto the M_{k+1}, that was set up during the training phase. The ϕ_k are nonlinear in general and dependent on a number of internal (learnable) parameters. These are usually called *weights* and are calibrated during training. As a result the manifolds M_k at each successive layer may be of very different dimensions.

2 Actual manifold dimensions

In [2] this exact situation was investigated for a well known image classifier algorithm possessing this type of layered neural network architecture. The authors estimated the *dimension*, D_k, of each of the M_k ($k = 1, 2, ..., K$) using the two-nearest-neighbour (twoNN) algorithm [3] over the sampled points. This yields the fractal (Hausdorff) dimension of M_k. Importantly it was shown [2] that although the embedding dimensions, m_k, were decreasing from the first layer to the final layer (by architectural design), the dimensions, D_k, were not always decreasing: for some transitions, from layer k to layer $k + 1$, it was possible that the dimension increased. Such a phenomenon occurs when M_k becomes folded over and over by ϕ_k and thus fills a manifold M_{k+1} of higher dimension (just as when we have familiar examples of space filling curves and volume filling surfaces).
Of course any over intricate folding, that we will refer to as "pleating", means that there must be some points within the image clouds that are not close together in M_k yet are mapped by ϕ_k onto image points in M_{k+1} which are very close together. The perturbation distance between such points thus becomes much smaller under ϕ_k, meaning that the local inverse must have a high gradient. Of course, when mapped back into input space, these perturbations were not represented and sampled within the input data object set, where they would be large and very possibly non-realistic deformations (not small data perturbations but large localised *semantic* perturbations).

3 Vulnerabilities

There is a possible relationship between this phenomenon and an algorithm's vulnerability to adversarial attacks, or *spoofing*, a growing threat for machine learning research, especially via *black box attacks* where the attacker has no access to the model's internal parameters (see the discussion and references in [2]). Typically AI classifiers are subject to input perturbations that appear negligible to humans yet cause misclassifications. Hence the observable, put perhaps un-anticipated, occurrence of dimensional layer-to-layer increases (most likely due to pleating) is perhaps to be avoided. Ironically, given developers' full access to the internal activations it is relatively inexpensive to make such calculations using the method in [3]. This could become a standard performance check.

4 Related research

There is a very similar approach taken in [4]. There the layer-by-layer activations, within each E_k, are subject to the "Mapper Algorithm" which decomposes those point clouds into clusters, and then outputs a skeleton representation of the underlying topology [5]. In particular, when this method is applied to large language models (and their subsequent specialised fine tuning), calibrated by ensembles of many sentences, it is shown that (within the later layers) the branched topology uncovered by the skeleton is highly related to the linguistic and contextual information (distinct semantics and usages) of the words [4]. Furthermore [4] shows how the skeleton becomes more precise (within certain applications) as the model is fine tuned. In general terms, the skeletons become more disconnected with many ambiguities (represented by branch points) resolved during the fine tuning. So we observe that the skeleton's geometric representation of the structure of the point clouds within E_k represents the latent structure of meanings, context, and usage within the language samples. This of course is a highly satisfactory state of affairs, and it encourages us to

look at further ways that the activation point clouds with various layers might be analysis and exploited or controlled to improve performance and safety.

5 Topological/geometrical analysis

The estimate of dimension used in [2] is a very blunt tool though. In this paper we will deploy a more sophisticated method: computational topological data analysis, more specifically persistent homology, and the calculation of persistence diagrams of different degrees (see [6] and the references therein for a general introduction). In doing so we hope to gain further insights into the shape of the high dimensional point clouds, and the manifolds they are sampling, and to suggest a topological *loss term* that might penalise undesirable behaviour (pleating and so on) due to the latent layer-to-layer mappings, and thus reduce the vulnerability of calibrated algorithm to attacks and other misbehaviour.
Most calibration methods set the internal parameters (which in turn define the mappings) by minimising the loss term over a training data set. The loss term is usually adjusted, or regularised, to prevent over-fitting (for the training data), so it may be possible to amend this to include a suitable penalty related to the internal topological structure of the manifolds. This may result in lower risk algorithms.

6 An underview

In this section we consider a specific example before generalising.

6.1 Data

A residual network (ResNet) is a type of directed acyclic network that has residual (or shortcut) connections that bypass some subsequences, called stacks, of the main network layers. Residual connections enable the parameter gradients to propagate more easily from the output layer to the earlier layers of the network, which makes it possible to train deeper networks. The increased network depth can result in higher accuracies on more difficult tasks.
The architecture of the type of the ResNet that we consider has been designed to ease training in very deep networks and improve on the earlier VGG and Imagenet models [7]. The architecture comprises stacks, each containing residual blocks, see Figure 1. The stack's architecture is based on the idea that if multiple nonlinear layers within the kth stack can asymptotically approximate a complicated function, $\phi_k(x)$ say, then it is equivalent to assuming they can also asymptotically approximate the residual function $\phi_k(x) - x$. So, rather than have the layers approximate $\phi_k(x)$, we let them approximate a residual function $F(x) = \phi_k(x) - x$, since the ease of learning might be rather different.
We selected particular observation layers within the whole at the end of the successive stacks, after x is added back to $F(x)$ (at the layers stack1block4relu3, stack2block3relu3, stack3block2relu3).
The size of the embedding dimensions, m_k, at the end of the blocks has to suit both the residual function and the identity as they are added. Note that a linear projection is used to make *shortcut* connection match the nonlinear residual connection in terms of the embedding dimension. For example the embedding dimension of input into the stack that ends stack1block4relu3 increases from 16384 (32x32x16) to 65536 (32x32x64) in both routes (see Table 1 below).
We consider a ResNet with 104 layers designed to classify CIFAR colour images (32 x 32 x 3 pixels) into 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). More details about the network and the data can be found in [8]. The point clouds considered at each chosen sequential layer are based on a sample of N =10,000 test CIFAR-10 images, each of which contains 32-by-32 pixels. The layers' embedding dimensions, m_k, and the corresponding twoNN estimate of dimension, D_k, of each point cloud are shown in Table 1.
Although the embedding dimensions, the m_k, can be very high for our chosen layers within such networks, the estimation of the manifold dimensions, the D_k, and our implementation of persistent homology (see next

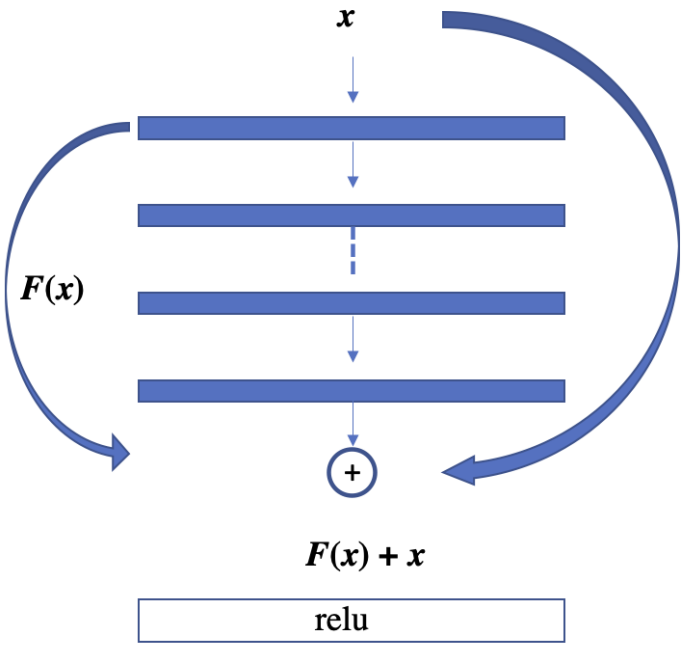

Figure 1: A single stack within a ResNet.

section) both merely require the pairwise distances between the points within the clouds within the E_k. This is also true of the Mapper algorithm investigation given in [4].
In estimating the dimensions, D_k, we tested for the size of possible sampling errors (when N =10,000) by taking 80% random sub-samples, recalculating, and examining the resulting distributions for each D_k. These were indeed significantly smaller than the layer-to-layer, differences, see Figure 2.

6.2 Persistent Homology

Persistent homology computes the different scales at which topological features, such as connected components, cyclic rings around holes, voids surrounded by closed two dimensional surfaces, and so on, are generated and extinguished within the point cloud (in this case at a single layer). A chosen real valued *filtration*, or scale parameter, is used to associate points pairwise within the point cloud. Then the resulting graph structure is analysed as that scale parameter increases from zero. In particular, the zeroth degree features, H_0, identify the various connected components of the partly connected point cloud; while the first degree features, H_1, identify the cyclic ring sub-structures around holes through the partly connected point cloud. In this application the scale parameter is always the distance between pairs of points within E_k at the kth layer. As the scale increases features within the whole cloud, such as a cyclic ring around a hole, will undergo birth (and first appear as a suitable subset of points become connected up) at some relatively small scale, b say, and then undergo death and become filled in (as points across the ring's diameter become connected) and disappear at some larger death scale, d say. We consider a feature to be *prominent* if the ratio d/b is relatively high (relatively to features that might be observed within point clouds of similar size drawn under some appropriate null model). In measuring the prominence of a single cycle we will take $\log(d/b)$, which is close to zero for non-prominent cycles (that are to be expected from sampling noise).
In Figure 3 we depict the *persistence diagrams*, plotting d versus b values for the H_1 (and H_0) persistent features for the point cloud at each layer. The absolute values of the scale parameter axes for each persistence diagram (that may be somewhat m_k, embedding, dependent) play no role here as we will only use d/b ratios. We hypothesise that local folding, or pleating, of the manifold M_k would decrease the prominence of noisy cycles as points in it become bunched up, so the surviving cycles are at a smaller scale and may be even subdivided. In Figure 4 we observe a decrease in the prominence sum over all cycles (weighted by log(death/birth))

Layer Name	Activation Type	Embedding Dimension m_k	Estimated Dimension, D_k
Input	None	3072	32.4
relu1	relu	16384	51.2
stack1block4relu3	relu	65536	93.5
stack2block3relu3	relu	32768	67.0
stack3block2relu3	relu	16384	44.9
global average pooling	pooling	256	16.2
fully connected	linear	10	8.1
softmax	softmax	10	2.8

Table 1: Eight layers selected from the example Residual Neural Network, showing each layer width (the embedding dimension) and the estimated dimension for the embedded manifolds, using the twoNN algorithm [3], sampled by the $N = 10,000$ point cloud.

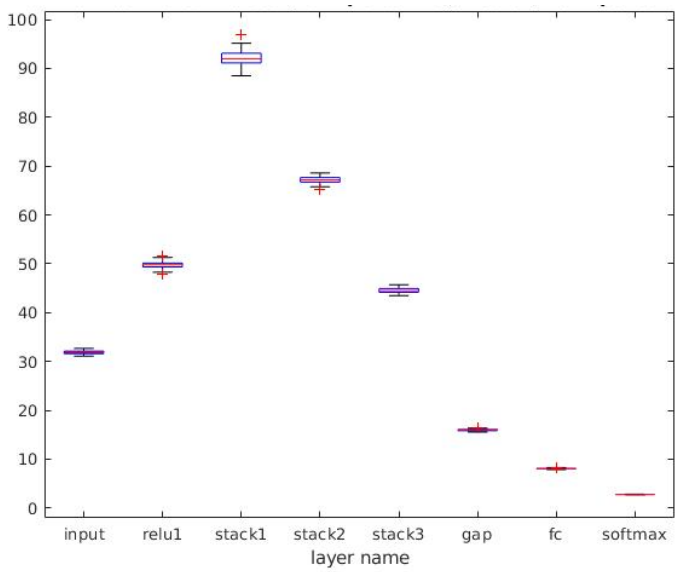

Figure 2: Box plots showing the possible sampling errors on the estimates of the dimension, D_k, by layer, given in Table 1, see [3]. We took 100 independent 80% random sub-samples in each case.

within each of the first two layers, which is consistent with the observed dimensional increases according to the above hypothesis. These two measures are highly related and negatively correlated (except for the final softmax layer).
In fact, the prominences of features (such as the H_1 cyclic features considered in Figure 4) have been recently discussed [9]. The observed cumulative distribution of the $\log(d/b)$ values obtained here, as shown layer by layer in Figure 5, follows the left-skewed Gumbel distribution (after zeroing the mean). This accords with empirical observations for many data sets [9].

7 Proactive regularisation

7.1 Possible topological regularisation

Our experiments suggest that a new *topological regularisation* term within the loss function, to be minimised during training, could be introduced to penalise combinations of weights that generate unwanted topological features in the data clouds for intermediate layers and the mappings between them. Ideally such a term

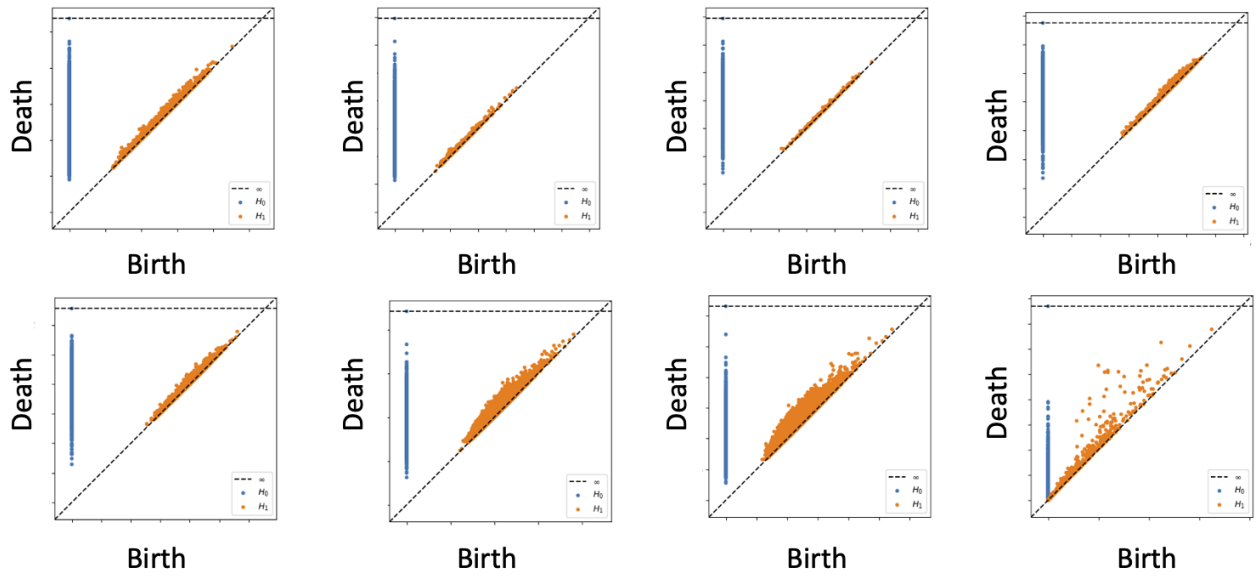

Figure 3: Persistence diagrams for each of eight layers. In each case the orange points indicate the (birth, death) scale coordinates of the corresponding persistent cyclic ring features (H_1), while the blue points indicate the scale coordinates of the separate persisting connected components of the point cloud (H_0).

would reduce the dimensional expansion as measured by an increase in the estimated dimension of the layer point clouds, under the continuous mapping, ϕ_k, that maps the M_k onto the M_{k+1}.
Such a novel topologically-aware approach to network training would contribute to an emerging field. Indeed, a recent review [10] considers *topological machine learning*, the intersection of topology-based methods and machine learning algorithms, and identifies some common threads and future challenges. It discusses *intrinsic* topological features that incorporate topological information directly into the design and regularisation of a machine learning model.
In [11] the authors propose a measure of topological complexity for the classification boundary of a given classifier that may be used for regularisation in order to force the topological complexity of the decision boundary to be simpler. That topological information acts as a penalty during classification, whilst differentiability, needed to enable minimisation, is obtained through the piece-wise linear approximation the regularised loss.
Some similar, more recent, work [12] has focused on smoothing the internal mappings by reducing the curvatures of decision boundaries (in terms of network parameters), and giving conditions for producing flat, or developable, decision boundaries.

7.2 Possible geometrical regularisation

A more direct regularisation approach is to consider the typical and relevant Jacobean for the continuous mapping, ϕ_k , that maps the M_k onto the M_{k+1}. At any point in the cloud sampling M_k, we have the $m_{k+1} \times m_k$ Jacobean (the linearization of ϕ_k, representing its first derivatives). This has $r_k = \min(m_k, m_{k+1})$ singular values in its singular value decomposition. Each represents a scaling factor relevant to suitable directions within E_k and E_{k_1}.
Clearly, in this application, each E_k has a natural size, since the kth layer has activations for each of the m_k artificial neurons, with each scaled to be in [0,1]: so $E_k = [0,1]^{m_k}$. Hence the singular values represent geometric stretching or shrinking in the various r_k orthogonal directions.
During the calibration of ϕ_k one should penalise large changes in the singular values for the Jacobean as we move through the point cloud inside E_k, which could represent folding or pleating. Such changes are

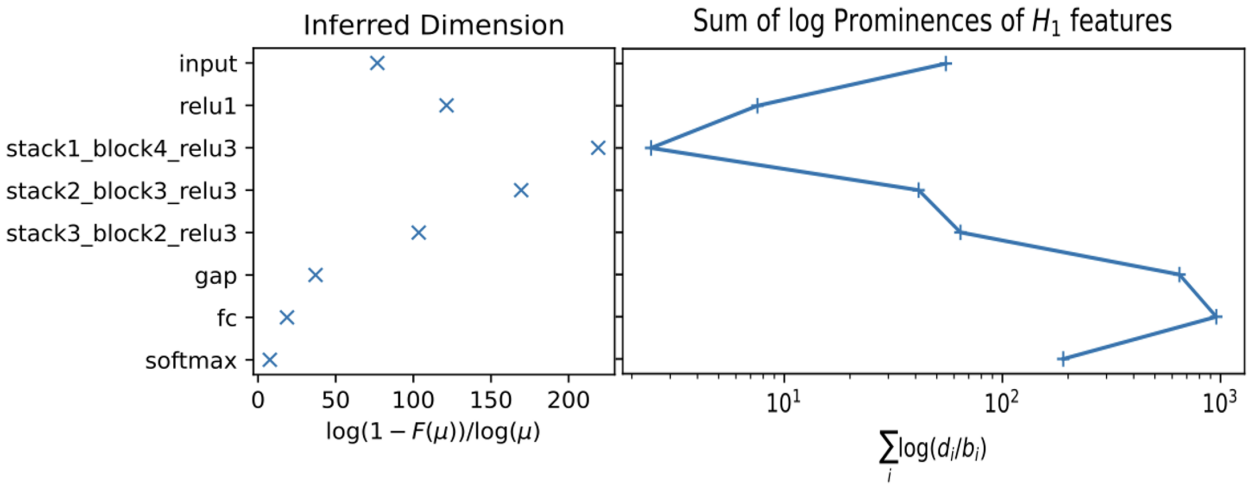

Figure 4: Left: the inferred dimension of the point cloud at successive layers (given by the estimator $\log(1-F(\mu))/\log(\mu)$, see [3]). Right: the sum of the log-prominences for cyclic (H_1) features (note that the short-lived cycles contribute almost zero).

represented by the $m_{k+1} \times m_k \rtimes (m_k+1)/2$ second derivatives of ϕ_k.
Of course any proposed explicit and smooth adjustment to the loss term can be usually dealt with by the gradient descent optimisation (a point noted in [13]). This factor alone makes a consideration of the Jacobean and higher derivatives attractive, as opposed to any non differentiable (yet topological) adjustments.

8 Discussion and further work

The main aim of this exploratory work is to illustrate that ideas from topological data analysis can be used to investigate the shape of the point clouds that pass through an AI system. The resulting insights shed light on the inner workings of the system; in particular how adversarial perturbations to input data can lead to misclassification. Building on work from [2], which applied a dimension estimation algorithm to the layered architecture VGG16, here we studied a more general RESNET architecture and incorporated a more sophisticated tool: persistent homology. In our tests we found that dimension estimation and persistent homology produce consistent summaries, with a decrease in the prominence of cycles accompanying an increase in the intrinsic dimension of the data cloud. It would be of interest to extend these studies to other network architectures and higher resolution images, and to develop a methodology for comparing and quantifying the properties of competing systems.
The results also led us to propose that a topological measure could be added to the loss function during training, as a means of encouraging beneficial properties, such as resilience to adversarial attack. The practical implementation of this type of topological regularization within real-world scenarios raises a number of challenges. Such a persistent homology pipeline, producing and analysing persistence diagrams during iterative training, is inherently non-differentiable and computationally expensive, even for the setting that we used here with smallish data clouds ($N = 10^4$) and parameter space ($\sim 10^7$). Indeed, the computational overhead associated with persistent homology calculation currently renders it exploratory rather than operational; so it may be viewed as a diagnostic tool. Further work is needed to focus on (i) the development of simpler *surrogate* measures, which may be more necessary than sufficient, for *pleating and folding* behaviour; and (ii) integrating such topological regularization into the training process.
We also note that studying the properties of feature space along a computational pipeline is currently an active topic in graph neural networks (GNNS). Here *oversmoothing* has been observed, where increasing network depth leads to homogeneous node representations—essentially a collapse of dimension [14]. Similarly, *over-squashing* refers to the circumstance where message passing bottlenecks arise, so that information fails

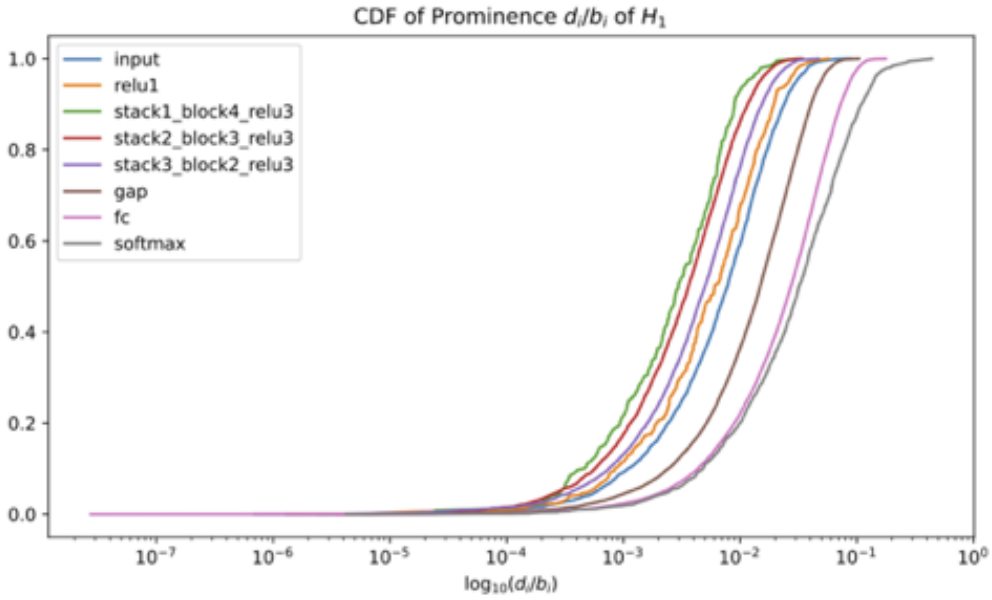

Figure 5: Left: layer by layer distributions of the $\log(d/b)$.

to propagate efficiently through a graph [15]. It would be of great interest to understand how topological data analysis techniques, of the type studied here, can be used to provide insights into these phenomena.

Funding and declarations

OG, PG, and HAH were funded by EPSRC grant EP/R018472/1. HAH gratefully acknowledges funding from a Royal Society University Research Fellowship. CFH was funded by EPSRC grants EP/T00097X/1 and EP/R018634/1. DJH was funded by EPSRC grant EP/V046527/1. KMY was funded by Oxford Mathematics and subsequently by UKRI grant MR/W01176X/1 (PI: J. Harvey).

We declare no conflicts of interest.

References

[1] Huang, X., Kwiatkowska, M., Wang, S. & Wu, M. (2017) Safety verification of deep neural networks. In: Majumdar, R., Kunčak, V. (eds) Computer Aided Verification. CAV 2017. Lecture Notes in Computer Science(), vol 10426. Springer, Cham. https://doi.org/10.1007/978-3-319-63387-9_1.

[2] Cui, Z., and Grindrod, P. (2022). Mappings, dimensionality and reversing out of deep neural networks, IMA Journal of Applied Mathematics, 2023;https://doi.org/10.1093/imamat/hxad019.

[3] Facco, E., d'Errico, M., Rodriguez, A., & Laio, A. (2017) Estimating the intrinsic dimension of datasets by a minimal neighborhood information. Sci Rep 7, 12140, https://doi.org/10.1038/s41598-017-11873-y.

[4] Rathore, A., Zhou, Y., Srikumar, V., & Wang, B. (2023). TopoBERT: Exploring the topology of fine-tuned word representations. Information Visualization, 22(3), 186–208 https://doi.org/10.1177/14738716231168671.

[5] Singh, G., Mémoli, F., and Carlsson, G. E. (2007). Topological methods for the analysis of high-dimensional data sets and 3D object recognition. In SPBG, pages 91–100.

[6] Otter, N., Porter, M.A., Tillmann, U., Grindrod, P., Harrington, H.A. (2027). A roadmap for the computation of persistent homology. EPJ Data Sci. 2017;6(1):17. doi: 10.1140/epjds/s13688-017-0109-5. Epub 2017 Aug 9. PMID: 32025466; PMCID: PMC6979512.

[7] He K., Z. Xiangyu, Z., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778.

[8] Mathworks, Train residual network for image classification, `https://uk.mathworks.com/help/deeplearning/ug/train-residual-network-for-image-classification.html`.

[9] Bobrowski, O., & Skraba, P. (2022). On the universality of random persistence diagrams, `https://arxiv.org/abs/2207.03926`.

[10] Hensel, F., Moor, M., & Rieck, B. (2021). A survey of topological machine learning methods. Front. Artif. Intell. 4:681108. doi: 10.3389/frai.2021.681108.

[11] Chen, C., Ni, X., Bai, Q., & Wang, Y. (2019). A topological regularizer for classifiers via persistent homology," in Proceedings of Machine Learning Research, eds K. Chaudhuri and M. Sugiyama (PMLR), 2573–2582.

[12] Liu, B., & Shen, M. (2022). Some geometrical and topological properties of DNNs' decision boundaries, Theoretical Computer Science, Volume 908, 2022, Pages 64-75, ISSN 0304-3975, `https://doi.org/10.1016/j.tcs.2021.11.013`.

[13] Patra, R., Hebbalaguppe, R., Dash, T., Shroff, G., & Vig, L. (2023). Calibrating deep neural networks using explicit regularisation and dynamic data pruning, in 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2023 pp. 1541-1549. doi: 10.1109/WACV56688.2023.00159.

[14] Wu, X., Ajorlou, A., Wu, Z., & Jadbabaie, A., Demystifying oversmoothing in attention-based Graph Neural Networks, Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS), 2023.

[15] Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., Bronstein, M., Understanding over-squashing and bottlenecks on graphs via curvature, Tenth International Conference on Learning Representations (ICLR,) 2022.