What does it mean to learn in deep networks? And, how does one detect adversarial attacks? Supplementary Material

Ciprian A. Corneanu, Univ. Barcelona
Meysam Madadi, CVC, UAB
Sergio Escalera, Univ. Barcelona
Aleix M. Martinez, OSU

1. Additional Results

The main paper illustrated the use of our approach on the LeNet, a historical DNN. In this section, we replicate those results on a recent DNN: VGG16 [2].

We used the exact same experimental setup of Section 3.8, including identical learning rate, weight decay, momentum, etc. The difference is that we used the CIFAR10 [1] dataset instead. CIFAR10 consists of 60,000 training images and 10,000 separate test images labelled with 10 different classes. These are natural images representing common objects (*e.g.*, dog, cat, airplane). For training, data was augmented using random horizontal flips and random crops of equal image size with padding $= 4$. All data was resized to 32×32 pixels and mean and variance were normalized.

Fig. 9 shows the results of our approach. The first row in this figure shows the number of 1D cavities as a function of the density of the functional binary graph representing VGG16. Different plots specify these number at distinct epochs. The second row does the same but for 2D cavities.

Note how the maximum number of 1D and 2D cavities (indicated by a dashed red vertical line) moves toward less dense functional representations of the VGG16, as described in the main paper and Algorithm 2. The lowest density is achieved at epoch 80. After that, there is a small regression toward higher densities, suggestive of overfitting. These results are confirmed by the plot of the testing accuracy shown in Fig. 10. In this figure we see that while the training accuracy keeps growing after epoch 80, the testing accuracy does not. Instead, the testing accuracy slightly decreases after epoch 80, as suggested by our approach.

In Fig. 11, we show the plots of 1D and 2D cavities for the original (unaltered) training samples (blue curve) and for the adversarial samples (red curve). The plots are given as a function of the density. As we can see in the figure, the results mimic those reported in the main paper, allowing us to detect the adversarial attack.

2. Cavity Formation: An example

In the main paper we derived a theory that relates cavity formation with generalization. These cavities are given by the binary graphs defining the correlation of activation of the nodes of the DNN.

To clarify the formation of these cavities, let us illustrate this on a selected set of nodes of the LeNet DNN we used in the main paper, Fig. 12.

In this figure, we can see the formation of 1D cavities first, then the formation of 2D cavities, and, finally, the formation of a 3D cavity (second to last binary graph). Note how these cavities are formed as the density of the binary graph is increased. Then, in the right-most graph, we see how further increasing the density eliminates all the cavities.

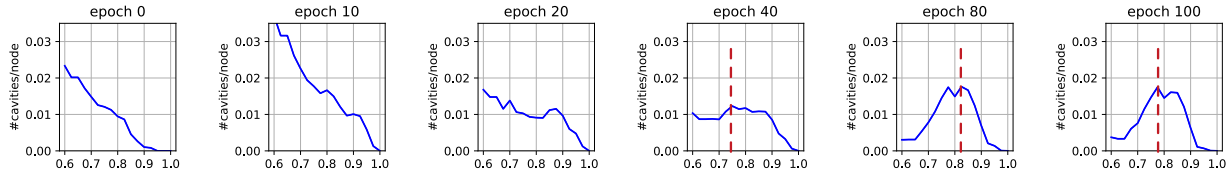
This particular example is for the activation graph formed at epoch 20. Layers are over-imposed to show location in the network. The location within the layer is chosen for the best visual effect. By increasing density (*i.e.*, decreasing T), more and more edges are added to the graph. Specifically, at $T = .8$, a couple of 2D cavities form. These contribute to an increase in β_1. At exactly $T = .63$ a 3D cavity is realized between these, resulting in an increase to β_2. This cavity is filled at higher densities ($T = .5$).

3. Training with Bettis

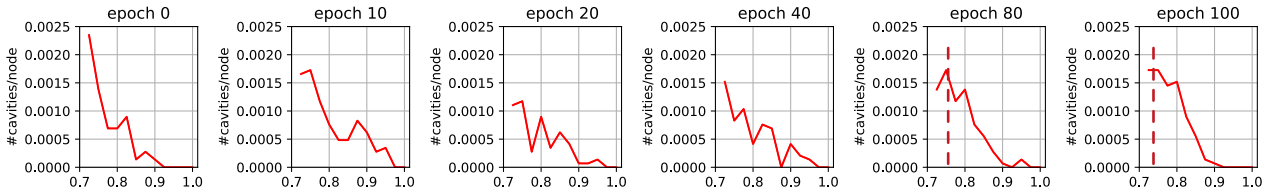
As the paper details, we can use our approach as a measure of generalization. Hence, Algorithm 2 can be used in lieu of the training or verification error to determine when the network has learned to generalize and before the network overfits to the training data.

This result is illustrated in Fig. 13. This figure shows the testing classification accuracy (in blue) and the difference of the peak densities at epoch t and $t-1$ (in black). Also note that classification accuracies are given on the left y-axis and values of $\Delta k = \hat{k}_t - \hat{k}_{t-1}$ are on the right y-axis.

When Δk is beyond a small threshold $-\epsilon$, Algorithm 2 decides to stop training. As shown in the figure that coin-



(a)



(b)

Figure 9. Results of the VGG16 on CIFAR. (a) Number of normalized 1D cavities (y-axis) as a function of graph density (x-axis) and number of epochs (given at the top of each plot). (b) Same as (a) but for 2D cavities.

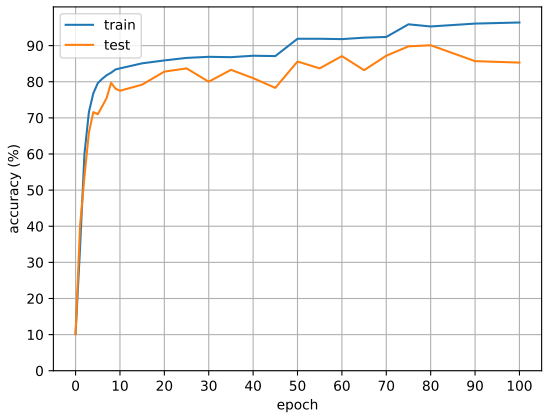


Figure 10. VGG16 training and testing accuracy on CIFAR10 as a function of training epochs.

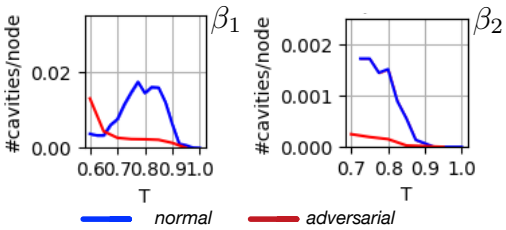


Figure 11. Betti numbers obtained when using unaltered and adversarial testing samples. VGG16 trained on CIFAR10.

cides with the point where generalization is achieved (thick red line in the figure). After that, the network starts to memorize the samples that do not generalize.

Specifically, given Δk, we can clearly observe three dynamic regimes for $\hat{k}$. Initially, in the first epochs $\hat{k}$ increases rapidly, it then reaches a maximum value between epochs

dimension	$n = 1$	$n \leq 3$
LeNet	.66	2.71
VGG16	5.15	20.24

Table 1. Time (in minutes) it takes to compute Betti numbers for LeNet and VGG16.

8 and 32 and then it starts decreasing in the later stage of training. Even where no verification set is available, one can use this knowledge to deduce when the training should stop. One can also imagine how this approach may one day be applied to unsupervised learning, but additional research will be necessary to make this a reality.

4. Algorithmic Complexity

Finally, we give a formal analysis of the computational complexity of the derived algorithm.

Let the binomial coefficient $p = \binom{N+1}{n+1}$ be the number of n-simplices of a simplicial complex S. In order to compute $\beta_n(S)$, one has to compute $rank(\partial_{n+1})$ where $\partial_{n+1} \in \mathbb{R}^{p \times q}$, p is the number of n-simplices, and q the number of $(n+1)$-simplices. This has polynomial complexity $O(q^a)$, $a > 1$.

Fortunately, the graphs of a network with N nodes are far from complete (*i.e.*, N-simplices). We take advantage of the graph sparsity to reduce computational complexity. This means that for typical DNNs, the number of n-simplices is way lower than the binomial coefficient defined above, unless one is interested in $\beta_n(S)$ for $n > 3$, or $\rho = 1$ (*i.e.*, $T = 0$).

In Table 1, we show the actual time (in minutes) it takes to compute cavities for a network with 1,820 nodes for $T > .5$. This corresponds to a single step of Algorithm 2, excluding training, on a single Intel Xeon, 2.2 GHz CPU.

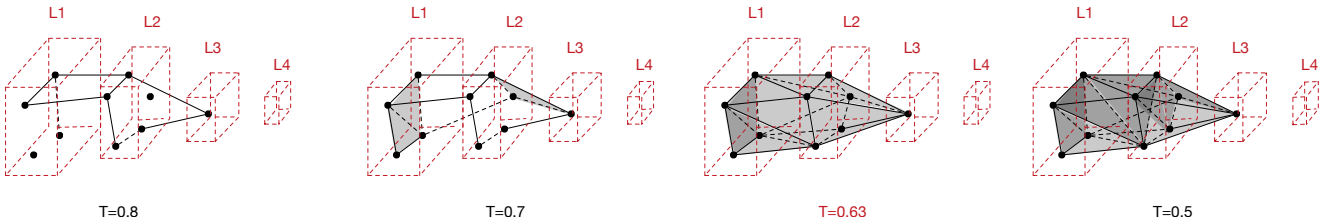


Figure 12. Example of 1D, 2D and 3D cavity formation in LeNet trained on MINST.

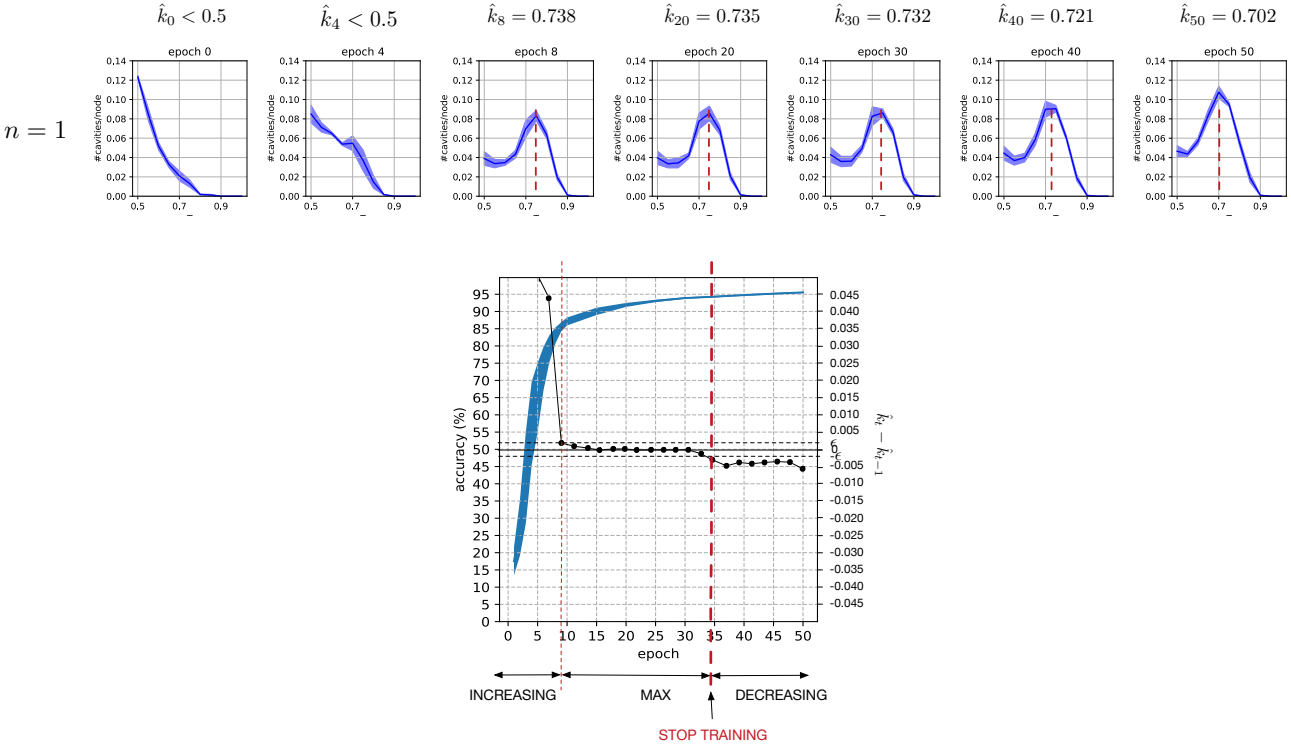


Figure 13. An illustration of Algorithm 2 for $n = 1$.

Acknowledgments

The authors and research presented in this paper were supported in part by the National Institutes of Health (NIH), grants R01-DC-014498 and R01-EY-020834, the Human Frontier Science Program (HFSP), grant RGP0036/2016, MINECO/FEDER TIN2016-74946-P, and the CERCA Programme / ICREA Academia program. We thank Litien Sun for his help.

Affiliations

Ciprian A. Corneanu is with the Department of Applied Mathematics and Analysis at the Universitat of Barcelona, Meysam Madadi is with the Computer Vision Center and the Universitat Autonoma de Barcelona, Sergio Escalera is with the Department of Applied Mathematics and Analysis at the Universitat de Barcelona, Aleix M. Martinez is with the Department of Electrical and Computer Engineering at The Ohio State University. CAC was a visiting scholar at OSU in 2018-2019. The research described in this paper was completed during CAC's time at OSU.

References

[1] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1