WHAT DOES IT MEAN TO LEARN IN DEEP NETWOKS? AND, HOW DOES ONE DETECT ADVERSARIAL ATTACKS?

CIPRIAN CORNEANU, MEYSAM MADADI, SERGIO ESCALERA, ALEIX MARTINEZ

UNIVERSITAT DE BARCELONA — CVC Centre de Visió per Computador — THE OHIO STATE UNIVERSITY

CVPR LONG BEACH CALIFORNIA June 16-20, 2019

PROBLEM

- Despite flexibility and high accuracy deep neural networks are increasingly opaque.
- Standard interpretability approaches do not generally allow us to determine where a DNN will succeed or fail and why.
- We derive a novel approach to define what it means to learn in DNNs and how to use this knowledge to detect adversarial attacks.

CONTRUBUTIONS

- We show that topological structure for networks that learn to generalize differs from those that simply memorize the training samples.
- This theoretical results allows us to derive an early-stopping algorithm (see Early Stopping using Topology) that does not require the use of a verification set.
- We show the same approach can be used to detect adversarial attacks.

MAIN THEORETICAL RESULTS

- **Learning** to generalize in DNN is defined by the creation of 2 and 3D cavities in the functional binary graphs representing the correlations of activation of distant nodes of the DNN, and the movement of 1D cavities from higher to lower graph density.
- **Memorizing** (overfitting) is indicated by a regression of these cavities toward higher densities in these functional binary graphs.

BETTI NUMBERS

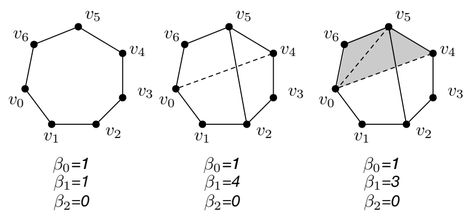


Figure 1. Betti numbers are sequences of natural numbers counting the cavities of a topological object in a corresponding dimension. Low dimensional Betti numbers have intuitive interpretation: β_0 counts connected components, β_1 counts 2D cavities, β_2 3D cavities.

MORE INFO

Scan me — *Paper*: https://cipriancorneanu.github.io/files/corneanu2019what.pdf

Scan me — *Code*: https://github.com/cipriancorneanu/dnn-topology

LEARNING AND MEMORIZATION

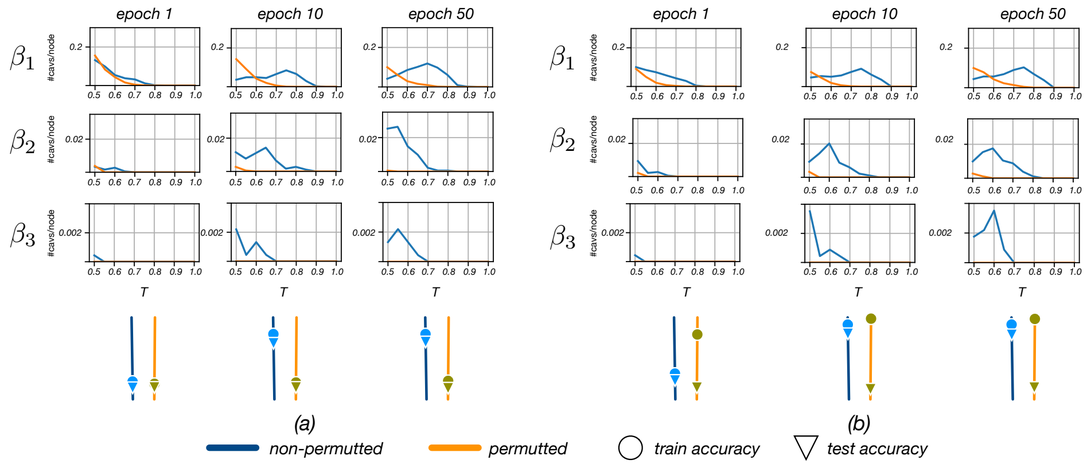


Figure 2. Betti numbers obtained when using 50% (left), and 1% (right) of the training data. Blue curves indicate non-permuted labels; orange curves indicate permuted labels. Training and testing accuracy indicated on the bottom row. Even though, high training accuracy is achieved, when labels are permuted this does not result in a concentration of cavities at low edge density indicating memorization, not learning.

LEARNING AND GENERALIZATION

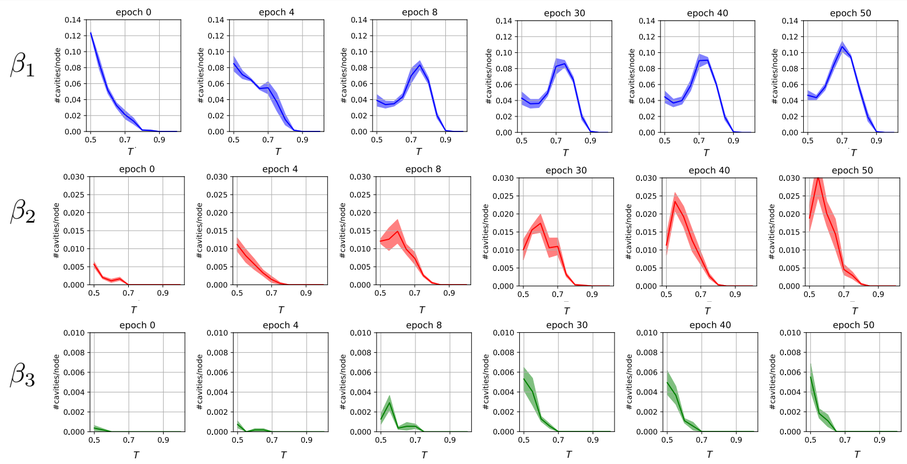
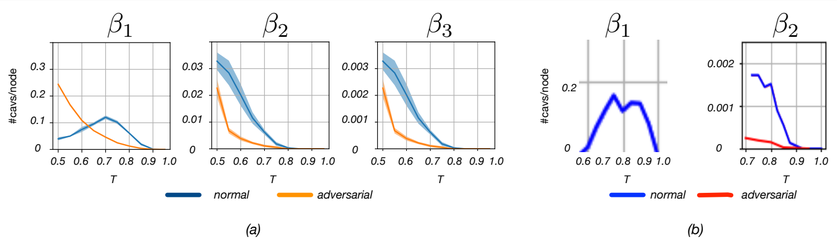


Figure 3. Evolution of Betti numbers of the LeNet network as a function of T and epochs. Recall, T is inversely proportional to edge density. The y-axis in the plots indicates the number of cavities properly normalized by the number of nodes (i.e., number of cavities/node). We hypothesize that learning in DNNs is equivalent to **finding the smallest density n-D cavities in the functional binary graphs that define the network**. Based on this, we propos a novel early stopping algorithm (see Algorithm 2).

DETECTING ADVERSARIAL ATTACKS

Figure 4. Betti numbers obtained when using unaltered and adversarial testing samples for Lenet on MNIST (a) and VGG16 on CIFAR10 (b). But for the data in the ad- versarial set, we expect only 1, 2 and 3D cavities at the highest densities, indicating local processing but a lack of global engagement of the network.



METHOD

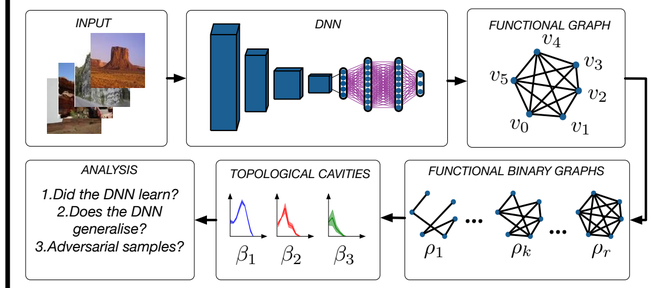


Figure 5. Approach proposed to study topological properties of a trained deep neural network.

Algorithm 1 Weighted to binary graph mapping.

1: Let $G = (V, E)$, $X = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$, and define $T > 0$.
2: Let $corr(G|X,T) = \{|c_{i_1 j_1}|, |c_{i_2 j_2}|, \ldots, |c_{i_r j_r}|\}$, s.t. $|c_{i_1 j_1}| \geq |c_{i_2 j_2}| \geq \cdots \geq |c_{i_r j_r}| > T$.
3: Let $G_0 = \emptyset$ and k=0.
4: **repeat**
5: $k = k + 1$.
6: $G_k = G_{k-1} \bigcup \{v_{i_k}, \hat{e}_{ij} v_{j_k}\}$, with $\hat{e}_{ij}$ the undirected edge defining v_{i_k}— v_{j_k}.
7: $\rho_k = k/r$.
8: **until** $k = r$.

EARLY STOPPING WITHOUT VERIFICATION DATA

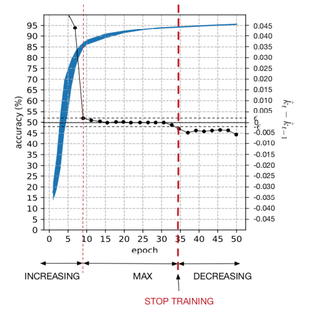


Figure 6. An illustration of Algorithm 2 for n=1 (left). We propose to use our approach as a measure of generalisation. Hence, Algorithm 2 can be used in lieu of the training or verification error to determine when the network has learned to generalize and before the network overfits (memorizes) the training data.

Algorithm 2 Generalization.

1: Let $G = (V, E)$ define a DNN with θ its parameters.
2: Let X be the training set, and set $T > 0$.
3: Set t=0, and n to either 1, 2, or 3.
4: **repeat**
5: Use X and the selected loss to optimize θ.
6: t=t+1.
7: Use Algorithm 1 to obtain G_k, $k = 1, \ldots, r$.
8: Compute the clique complexes S_k of G_k.
9: $\widehat{k}_t = \arg\max_k \beta_n(S_k)$.
10: **until** $\widehat{k}_t > \widehat{k}_{t-1}$.