

Computing the Testing Error without a Testing Set

Ciprian A. Corneanu
Univ. Barcelona, Tawny

Sergio Escalera
Univ. Barcelona, CVC

Alex M. Martinez
OSU, Amazon

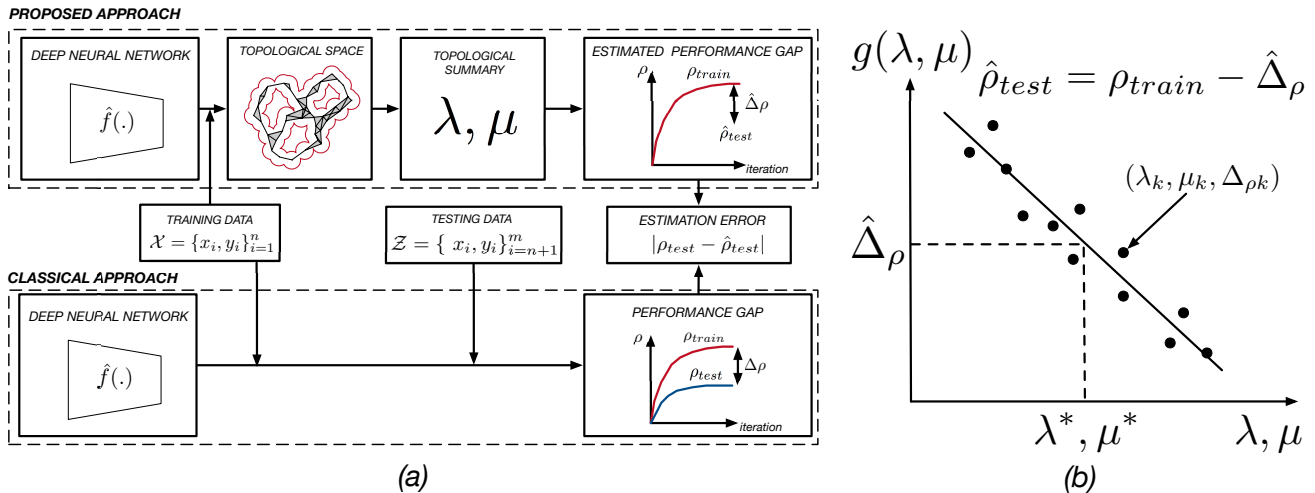


Figure 1: (a) We compute the test performance¹ of any Deep Neural Network (DNN) on any computer vision problem using **no** testing samples (top); neither labelled nor unlabelled samples are necessary. This is in sharp contrast to the classical computer vision approach, where model performance is calculated using a curated test dataset (bottom). (b) The persistent algebraic topological summary (λ^*, μ^*) given by our algorithm (x -axis) against the performance gap $\Delta\rho$ between training and testing performance (y -axis).

Abstract

Deep Neural Networks (DNNs) have revolutionized computer vision. We now have DNNs that achieve top (performance) results in many problems, including object recognition, facial expression analysis, and semantic segmentation, to name but a few. The design of the DNNs that achieve top results is, however, non-trivial and mostly done by trial-and-error. That is, typically, researchers will derive many DNN architectures (i.e., topologies) and then test them on multiple datasets. However, there are no guarantees that the selected DNN will perform well in the real world. One can use a testing set to estimate the performance gap between the training and testing sets, but avoiding overfitting-to-the-testing-data is almost impossible. Using a sequestered testing dataset may address this problem, but this requires a constant update of the dataset, a very expensive venture. Here, we derive an algorithm to estimate the performance gap between training and testing that does not require any testing dataset. Specifically, we derive a number of persistent topology measures that identify when a DNN is learning to generalize to unseen samples. This allows us to compute the DNN's testing error on unseen samples, even when

we do not have access to them. We provide extensive experimental validation on multiple networks and datasets to demonstrate the feasibility of the proposed approach.

1. Introduction

Deep Neural Networks (DNNs) are algorithms capable of identifying complex, non-linear mappings, $f(\cdot)$, between an input variable \mathbf{x} and an output variable \mathbf{y} , i.e., $f(\mathbf{x}) = \mathbf{y}$ [19]. Each DNN is defined by its unique topology and loss function. Some well-known models are [25, 15, 18], to name but a few.

Given a well-curated dataset with n samples, $\mathcal{X} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, we can use DNNs to find an estimate of the functional mapping $f(\mathbf{x}_i) = \mathbf{y}_i$. Let us refer to the estimated mapping function as $\hat{f}(\cdot)$. Distinct estimates, $\hat{f}_j(\cdot)$, will be obtained when using different DNNs and datasets. Example datasets we can use to this end are [10, 13, 11], among many others.

¹Let ρ_{train} and ρ_{test} be the performance of an algorithm computed using a training and testing set, respectively; $\hat{\rho}_{test}$ is the estimated testing error computed without any testing data. The performance metric may be classification accuracy, F1-score, Intersection-over-Union (IoU), etc.

Using datasets such as these to train DNNs has been very fruitful. DNNs have achieved considerable improvements in a myriad of, until recently, very challenging tasks, e.g., [18, 27].

Unfortunately, we do not generally know how the estimated mapping functions $\hat{f}_j(\cdot)$ will perform in the real world, when using independent, unseen images.

The classical way to address this problem is to use a testing dataset, Figure 1(a, bottom). The problem with this approach is that, in many instances, the testing set is visible to us, and, hence, we keep modifying the DNN topology until it works on this testing dataset. This means that we overfit to the testing data and, generally, our algorithm may not be the best for truly unseen samples.

To resolve this issue, we can use a sequestered dataset. This means that a third-party has a testing dataset we have never seen and we are only able to know how well we perform on that dataset once every several months. While this does tell us how well our algorithm performs on previously unseen samples, we can only get this estimate sporadically. And, importantly, we need to rely on someone else maintaining and updating this sequestered testing set. Many such sequestered datasets do not last long, because maintaining and updating them is a very costly endeavour.

In the present paper, we introduce an approach that resolves these problems. Specifically, we derive an algorithm that gives an accurate estimate of the performance gap between our training and testing error, *without the need of any testing dataset*, Figure 1(a, top). That means we do not need to have access to any labelled or unlabelled data. Rather, our algorithm will give you an accurate estimate of the performance of your DNN on independent, unseen sample.

Our key idea is to derive a set of topological summaries measuring persistent topological maps of the behavior of DNNs across computer vision problems. Persistent topology has been shown to correlate with generalization error in classification [9], and as a method to theoretically study and explain DNNs' behavior [6, 9, 28]. The hypothesis we are advancing is that the generalization gap is a function of the inner-workings of the network, here represented by its functional topology and described through topological summaries. We propose to regress this function and use it to estimate test performance based only on training data.

Figure 1(b) shows an example. In this plot, the x -axis shows a linear combination of persistent topology measures of DNNs. The y -axis in this plot is the value of the performance gap when using these DNNs on multiple computer vision problems. As can be seen in this figure, there is a linear relationship between our proposed topological summaries and the DNN's performance gap. This means that knowing the value of our topological summaries is as good as knowing the performance of the DNN on a sequestered dataset, but without any of the drawbacks mentioned above

– no need to depend on an independent group to collect, curate, and update a testing set.

We start with a set of derivations of the persistent topology measures we perform on DNNs (Section 2), before using this to derive our algorithm (Section 3). We provide a discussion of related work (Section 4) and extensive experimental evaluations on a variety of DNNs and computer vision problems, including object recognition, facial expression analysis, and semantic segmentation (Sections 5 and 6).

2. Topological Summaries

A DNN is characterized by its *structure* (i.e., the way its computational graph is defined and trained), and its *function* (i.e., the actual values its components take in response to specific inputs). We focus here on the latter.

To do this, we define DNNs on a topological space. A set of compact descriptors of this space, called *topological summaries*, are then calculated. They measure important properties of the network's behaviour. For example, a summary of the *functional topology* of a network can be used to detect overfitting and perform early-stopping [9].

Let A be a set. An *abstract simplicial complex* S is a collection of vertices denoted $V(A)$, and a collection of subsets of $V(A)$ called *simplices* that is closed under the subset operation, i.e., if $\sigma \in \psi$ and $\psi \in A$, then $\sigma \in A$.

The dimension of a simplex σ is $|\sigma| - 1$, where $|\cdot|$ denotes cardinality. A simplex of dimension n is called a n -simplex. A 0-simplex is realized by a single vertex, a 1-simplex by a line segment (i.e., an edge) connecting two vertices, a 2-simplex is the filled triangle that connects three vertices, etc.

Let $M = (A, \nu)$ be a metric space – the association of the set A with a metric ν . Given a distance ϵ , the Vietoris-Rips complex [26] is an abstract simplicial complex that contains all the simplices formed by all pairs of elements $a_i, a_j \in A$ with

$$\nu(a_i, a_j) < \epsilon, \tag{1}$$

for some small $\epsilon > 0$, and $i \neq j$.

By considering a range of possible distances, $E = \{\epsilon_0, \epsilon_1, \dots, \epsilon_k, \dots, \epsilon_n\}$, where $0 < \epsilon_0 \leq \epsilon_1 \leq \dots \leq \epsilon_k \leq \dots \leq \epsilon_n$, a Vietoris-Rips filtration yields a collection of simplicial complexes, $\mathcal{S} = \{S_0, S_1, \dots, S_k, \dots, S_n\}$, at multiple scales, Figure 2 [14].

We are interested in the *persistent topology properties* of these complexes across different scales. For this, we compute the p^{th} – persistent homology groups and the Betti numbers β_p , which gives us the ranks of those groups [14]. This means that the Betti numbers compute the number of *cavities* of a topological object.²

²Two objects are topologically equivalent if they have the same number of cavities (holes) at each of their dimensions. For example, a donut and a coffee mug are topologically equivalent, because each has a single 2D

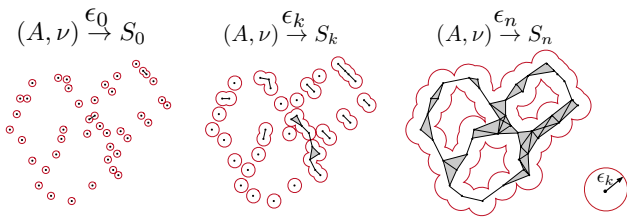


Figure 2: Given a metric space, the Vietoris-Rips filtration creates a nested sequence of simplicial complexes by connecting points situated closer than a predefined distance ϵ . Varying ϵ , we compute persistent properties (cavities) in these simplices. We define a DNN in one such topological space to compute informative data of its behaviour that correlates with its performance on testing data.

In DNNs, we can, for example, study how its functional topology varies during training as follows (Fig. 3). First, we compute the correlation of every node in our DNN to every other node at each epoch. Nodes that are highly correlated (*i.e.*, their correlation is above a threshold) are defined as connected, even if there is no actual edge or path connecting them in the network’s computational graph. These connections define a simplicial complex, with a number of cavities. These cavities are given by the Betti numbers. We know that the dynamics of low-dimension Betti numbers (*i.e.*, β_1 and β_2) is informative over the bias-variance problem (*i.e.*, the generalization vs. memorization problem) [9]. Similarly, it has been shown that these persistence homology measures can be used to study and interpret the data as points in a functional space, making it possible to learn and optimize the estimates $f_j(\cdot)$ defined on the data [6].

3. Algorithm

Recall $\mathcal{X} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ is the set of labeled training samples, with n the number of samples.

Let a_i be the activation value of a particular node in our DNN for a particular input \mathbf{x}_i . Passing the sample vectors \mathbf{x}_i through the network ($i = 1, \dots, n$), allows us to compute the correlation between the activation (a_p, a_q) of each pair of nodes which defines the metric ν of our Vietoris-Rips complex. Formally,

$$\nu_{pq} = \sum_{i=1}^n \frac{(a_{pi} - \bar{a}_p)(a_{qi} - \bar{a}_q)}{\varsigma_{a_p} \varsigma_{a_q}}, \quad (2)$$

where \bar{a} and ς_a indicate the mean and standard deviation over \mathcal{X} .

We represent the results of our persistent homology using a *persistence diagram*. In our persistence diagram, each

cavity, the whole in the donut and in the handle of the mug. On the other hand, a torus (defined as the product of two circles, $S^1 \times S^1$) has two holes because it is hollow. Hence, a torus is topologically different to a donut and a coffee mug.

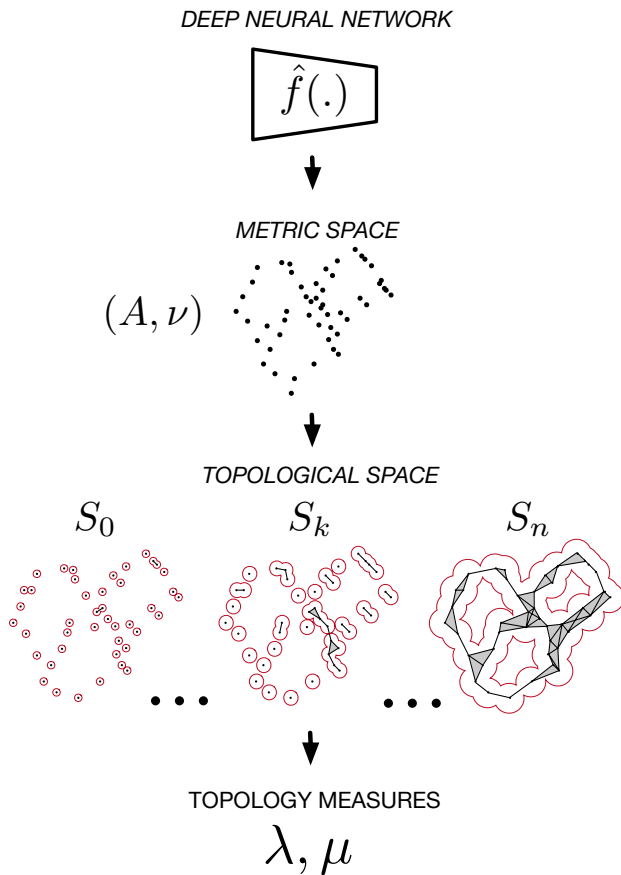


Figure 3: An overview of computing topological summaries from DNNs. We first define a set of nodes A in the network. By computing the correlations between these nodes we project the network into a metric space (A, ν) from which we obtain a set of simplicial complexes in a topological space through Vietoris-Rips filtration. Persistent homology on this set of simplicial complexes results in a persistence diagram from which topological measures can be computed directly.

point has as coordinates a set of pairs of real positive numbers $P = \{(\epsilon_d^i, \epsilon_b^i) | (\epsilon_d^i, \epsilon_b^i) \in \mathbb{R} \times \mathbb{R}, i = 1, \dots, C\}$, where the subscripts b and d in ϵ indicate the birth and death distances of a cavity i in the Vietoris-Rips filtration, and C is the total number of cavities.

A filtration of a metric space (A, ν) is a nested subsequence of complexes that abide by the following rule: $S_0 \subseteq S_1 \subseteq \dots \subseteq S_n = \mathcal{S}$ [30]. Thus, this filtration is in fact what defines the persistence diagram of a k -dimensional homology group. This is done by computing the creation and deletion of k -dimensional homology features. This, in turn, allows us to compute the lifespan homological feature [7].

Based on this persistence diagram, we define the *life* of a cavity as the average time (*i.e.*, persistence) in this diagram.

Algorithm 1 Computes the function $g(\lambda_k, \mu_k; \mathbf{c})$ that maps the topological summaries to the estimated testing error.

Input: train dataset $\mathcal{X} = \{x_i, y_i\}_{i=1}^n$, test dataset $\mathcal{Z} = \{x_i, y_i\}_{i=n+1}^m$.

Output: parametrized function g .

Parameters: $K = \#\text{trainings}$.

for $k=1 \dots K$ **do**

$\omega \leftarrow \arg \min_{\omega} L(\hat{f}(x; \omega), y)$.

▷ Train DNN and estimate f by optimizing loss L over \mathcal{X} .

$\Delta_{\rho k} = \rho_{\text{train}} - \rho_{\text{test}}$

▷ Compute performance gap.

for all pairs of nodes $(a_p, a_q) \in A, p \neq q$ **do**

$\nu_{pq} \leftarrow \sum_{i=1}^n \frac{(a_{pi} - \bar{a}_p)(a_{qi} - \bar{a}_q)}{s_{ap} s_{aq}}$

▷ Compute correlations. (Eq. 2).

end for

$\mathcal{S} \leftarrow VR(A, \nu)$

▷ Perform Vietoris-Rips filtration and get set of simplicial complexes \mathcal{S} .

$P \leftarrow \mathcal{PH}(\mathcal{S})$

▷ Compute persistent homology \mathcal{PH} over \mathcal{S} and get persistent diagram P .

$\lambda_k \leftarrow \frac{1}{C} \sum_{i=1}^C (\epsilon_d^i - \epsilon_b^i)$

▷ Compute topological summary (Eq. 3).

$\mu_k \leftarrow \frac{1}{C} \sum_{i=1}^C \frac{\epsilon_d^i + \epsilon_b^i}{2}$

▷ Compute topological summary (Eq. 4).

end for

$\mathbf{c} \leftarrow \arg \min_{\mathbf{c}} \sum_{i=1}^K (\Delta_{\rho k} - g(\lambda_k, \mu_k; \mathbf{c}))^2$

▷ Compute g by regression of $\{\lambda_k, \mu_k, \Delta_{\rho k}\}_{k=1}^K$

Formally,

$$\lambda = \frac{1}{C} \sum_{i=1}^C (\epsilon_d^i - \epsilon_b^i). \quad (3)$$

Similarly, we define its *midlife* as the average density of its persistence. Formally,

$$\mu = \frac{1}{C} \sum_{i=1}^C \frac{\epsilon_d^i + \epsilon_b^i}{2}. \quad (4)$$

Finally, we define the *linear* functional mapping from these topological summaries to the gap between the training and testing error as,

$$g(\lambda, \mu; \mathbf{c}) = \hat{\Delta}\rho, \quad (5)$$

where $\hat{\Delta}\rho$ is our estimate of the gap between the training and testing errors, and $g(\lambda, \mu; \mathbf{c}) = c_1\lambda + c_2\mu + c_3$, with $c_i \in \mathbb{R}^+$, and $\mathbf{c} = (c_1, c_2, c_3)^T$, Figure 1(b).

With the above result we can estimate the testing error without the need of any testing data as,

$$\hat{\rho}_{\text{test}} = \rho_{\text{train}} - \hat{\Delta}\rho, \quad (6)$$

where ρ_{train} is the training error computed during training with \mathcal{X} .

Given an actual testing dataset $\mathcal{Z} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=n+1}^m$, we can compute the accuracy of our estimated testing error as,

$$\text{Error} = |\rho_{\text{test}} - \hat{\rho}_{\text{test}}|, \quad (7)$$

where ρ_{test} is the testing error computed on \mathcal{Z} .

The pseudo-code of our proposed approach is shown in Alg. 1.³

³Code available at <https://github.com/cipriancorneanu/dnn-topology>.

3.1. Computational Complexity

Let the binomial coefficient $p = \binom{N+1}{n+1}$ be the number of n -simplices of a simplicial complex S (as, for example, would be generated during the Vietoris-Rips filtration illustrated in Fig. 2). In order to compute persistent homology of order n on S , one has to compute $\text{rank}(\partial_{n+1})$, with $\partial_{n+1} \in \mathbb{R}^{p \times q}$, p the number of n -simplices, and q the number of $(n+1)$ -simplices. This has polynomial complexity $O(q^a)$, $a > 1$.

Fortunately, in Alg. 1, we only need to compute persistent homology of the first order. Additionally, the simplicial complexes generated by the Vietoris-Rips filtration are generally extremely sparse. This means that for typical DNNs, the number of n -simplices is way lower than the binomial coefficient defined above. In practice, we have found 10,000 to be a reasonable upper bound for the cardinality of A . This is because we define nodes by taking into account structural constraints on the topology of DNNs. Specifically, a node a_i is a random variable with value equal to the mean output of the filter in its corresponding convolutional layer. Having random variables allows us to define correlations and metric spaces in Alg. 1. Empirically, we have found that defining nodes in this way is robust, and similar characteristics, e.g. high correlation, can be found even if a subset of filters is randomly selected. For smaller, toy networks there is previous evidence [9] that supports that functional topology defined in this way is informative for determining overfitting in DNNs.

Finally, the time it takes to compute persistent homology, and consequently, the topological summaries, λ and μ , is 5 minutes and 15 seconds for VGG16, one of the most extended networks in our analysis. This corresponds to a single iteration of Alg. 1 (the for-loop that iterates over k),

excluding training, on a single 2.2 GHz Intel Xeon CPU.

4. Related Work

Topology measures have been previously used to identify over-fitting in DNNs. For example, using lower dimensional Betti curves (which calculates the cavities) of the functional (binary) graph of a network [9], which can be used to perform early stopping in training and detect adversarial attacks. Other topological measures, this time for characterizing and monitoring *structural* properties, have been used for the same purpose [24].

Other works tried to address the crucial question of how the generalization gap can be predicted from training data and network parameters [3, 2, 23, 16]. For example, a metric based on the ratio of the margin distribution at the output layer of the network and a spectral complexity measure related to the network’s Lipschitz constant has been proposed [4]. In [23], the authors developed bounds on the generalization gap based on the product of norms of the weights across layers. In [2], the authors developed bounds based on noise stability properties of networks showing that more stability implies better generalization. And, in [16], the authors used the notion of margin in support vector machines to show that the normalized margin distribution across a DNN’s layers is a predictor of the generalization gap.

5. Experimental Settings

We have derived an algorithm to compute the testing accuracy of a DNN that does not require access to any testing dataset. This section provides extensive validation of this algorithm. We apply our algorithm in three fundamental problems in computer vision: object recognition, facial action unit recognition, and semantic segmentation, Figure 4.

5.1. Object Recognition

Object recognition is one of the most fundamental and studied problems in computer vision. Many large scale databases exist, allowing us to provide multiple evaluations of the proposed approach.

To give an extensive evaluation of our algorithm, we use four datasets: CIFAR10 [17], CIFAR100 [17], Street View House Numbers (SVHN) [22], and ImageNet [10]. In the case of ImageNet, we have used a subset with roughly 120,000 images split in 200 object classes [1].

We evaluate the performance of several DNNs by computing the classification accuracy, namely the number of predictions the model got right divided by the total number of predictions, Figure 5 and Tables 1 & 2.

5.2. Facial Action Unit Recognition

Facial Action Unit (AU) recognition is one of the most difficult tasks for DNNs, with humans significantly over-



Figure 4: We evaluate the proposed method on three different vision problems. (a) Object recognition is a standard classification problem consisting in categorizing objects. We evaluate on datasets of increasing difficulty, starting with real world digit recognition and continuing towards increasingly challenging category recognition. (b) AU recognition involves recognizing local, sometimes subtle patterns of facial muscular articulations. Several AUs can be present at the same time, making it a multi-label classification problem. (c) In semantic segmentation, one has to output a dense pixel categorization that properly captures complex semantic structure of an image.

performing even the best algorithms [3, 5].

Here, we use BP4D [29], DISFA [21], and EmotionNet [13], of which, in this paper, we use a subset of 100,000 images. And, since this is a binary classification problem, we are most interested in computing precision and recall, Figure 6 and Tables 3 & 4.

5.3. Semantic Segmentation

Semantic segmentation is another challenging problem in computer vision. We use Pascal-VOC [12] and Cityscapes [8]. The version of Pascal-VOC used consists of 2,913 images, with pixel based annotations for 20 classes. The Cityscapes dataset focuses on semantic understanding of urban street scenes [8]. It provides 5,000 images with dense pixel annotations for 30 classes.

Semantic segmentation is evaluated using union-over-intersection (IoU⁴), Figure 7 and Table 5.

⁴Also known as the Jaccard Index., which counts the number of pixels common between the ground truth and prediction segmentation masks divided by the total number of pixels present across both masks.

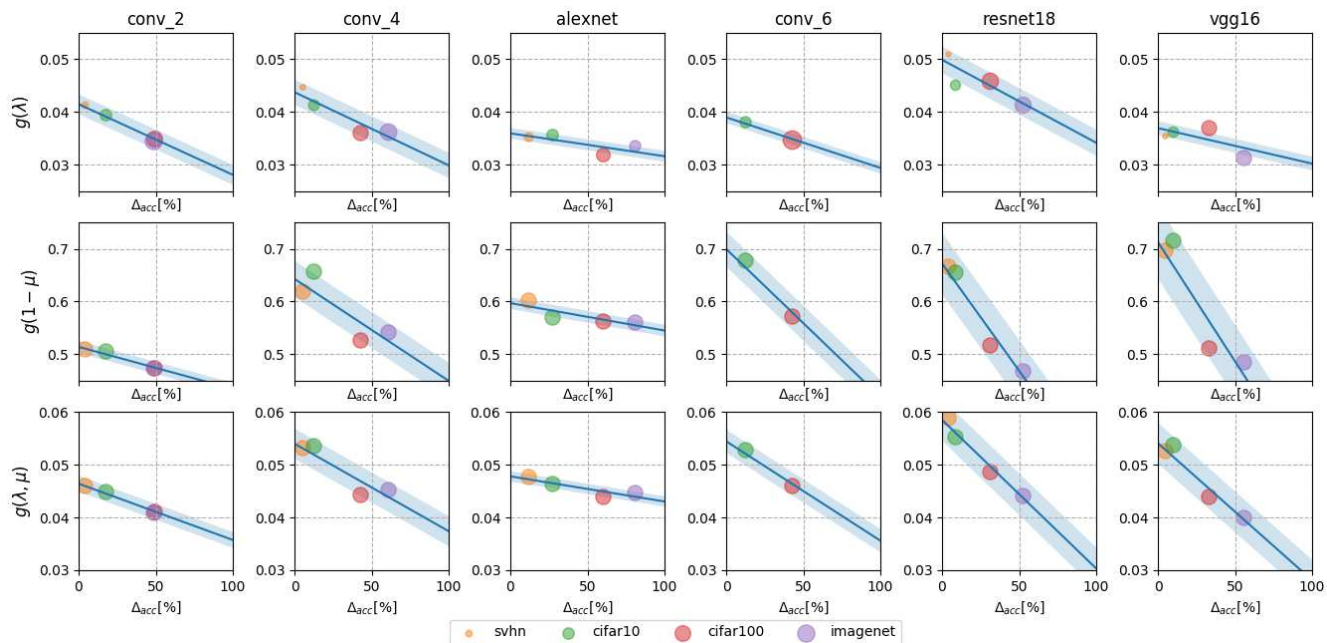


Figure 5: Topology summaries against performance (accuracy) gap for different models trained to recognize objects. Each disc represents mean (centre) and standard deviation (radius) on a particular dataset. Linear mapping and the corresponding standard deviation of the observed samples are marked.

5.4. Models

We have chosen a wide range of architectures (*i.e.*, topologies), including three standard and popular models [18, 25, 15] and a set of custom-designed ones. This provides diversity in depth, number of parameters, and topology. The custom DNNs are summarized in Table 6.

For semantic segmentation, we use a custom architecture called Fully Convolutional Network (FCN) capable of producing dense pixel predictions [20]. It casts classical classifier networks [15, 25] into an encoder-decoder topology. The encoder can be any of the networks previously used.

5.5. Training

For all the datasets we have used, if a separate test set is provided, we attach it to the train data and perform a k -fold cross validation on all available data. For object recognition $k = 10$, and for the other two problems $k = 5$. Each training is performed using a learning rate $\in \{0.1, 0.01, 0.001\}$ with random initialization. We also train each model on 100%, 50% and 30% of the available folds. This increases the generalization gap variance for a specific dataset. In the results presented below, we show all the trainings that achieved a performance metric above 50%. We skip extreme cases of generalization gaps either close to maximum (95%).

For object recognition the input images are resized to 32×32 color images, unless explicitly stated. Also they

are randomly cropped and randomly flipped. In the case of semantic segmentation all input images are 480×480 color images. No batch normalization (except for ResNet which follows the original design), dropout, or other regularization techniques are used during training. We train with a sufficient fixed number of epochs to guarantee saturation in both training and validation performance.

We use a standard stochastic gradient descent (SGD) optimizer for all training with momentum = .9 and learning rate and weight decay as indicated above. The learning rate is adaptive following a plateau criterion on the test performance, reducing to a quarter every time the validation performance metric does not vary outside a range for a fixed number of epochs.

6. Results and Discussion

Topological summaries are strongly correlated with the performance gap. This holds true over different vision problems, datasets and networks.

Life λ , the average persistence of cavities in the Vietoris-Rips filtration, is negatively correlated with the performance gap, with an average correlation of -0.76 . This means that the more structured the functional metric space of the DNN (*i.e.*, larger wholes it contains), the less it overfits.

Midlife μ is positively correlated with the performance gap, with an average correlation of 0.82 . *Midlife* is an indicator of the average distance ϵ about which cavities are

Model	conv_2	conv_4	alexnet	conv_6	resnet18	vgg16	mean
$g(\lambda)$	7.54 ± 5.54	10.12 ± 6.93	18.85 ± 11.36	6.70 ± 4.25	11.98 ± 8.28	16.01 ± 8.00	11.86 ± 7.39
$g(\mu)$	7.98 ± 7.95	12.69 ± 5.59	15.87 ± 10.25	7.08 ± 6.01	6.79 ± 4.65	8.93 ± 5.47	9.89 ± 6.65
$g(\lambda, \mu)$	7.62 ± 5.88	9.45 ± 6.88	15.26 ± 7.94	6.57 ± 5.80	4.92 ± 3.50	6.93 ± 5.10	8.45 ± 5.85

Table 1: Evaluation of Alg. 1 for object recognition. Mean and standard deviation error (in %) in estimating the test performance with leave-one-sample-out.

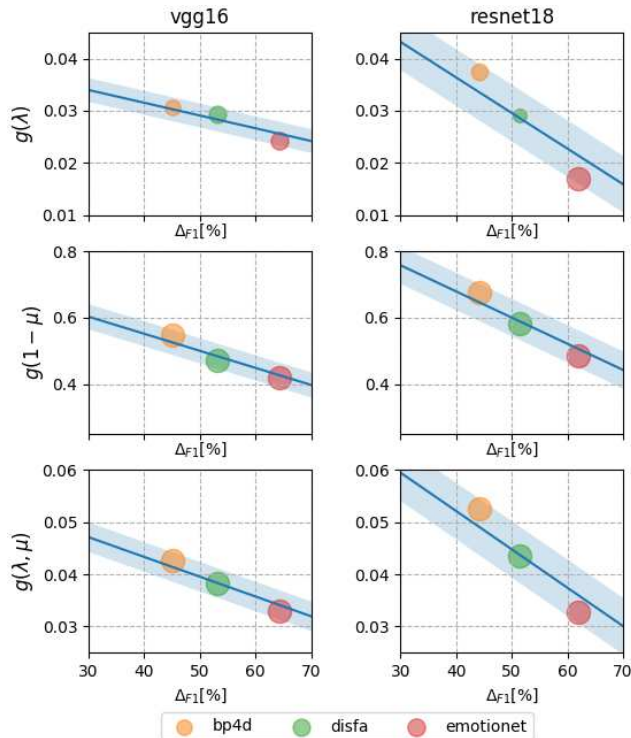


Figure 6: Topology summaries against performance (F1-score) gap for different models trained for AU recognition. Each disc represents mean (centre) and standard deviation (radius) for a 10-fold cross validation. Linear mapping and the corresponding standard deviation of the observed samples are marked.

Model	conv2	conv4	vgg16	resnet18
svhn	10.94 ± 4.99	10.73 ± 2.60	8.01 ± 2.31	11.09 ± 4.18
cifar10	9.64 ± 3.60	5.24 ± 1.92	9.41 ± 1.72	5.67 ± 0.73
cifar100	4.79 ± 6.20	22.46 ± 6.87	10.93 ± 1.46	6.71 ± 1.51
imagenet	8.33 ± 5.48	21.84 ± 7.35	13.49 ± 7.14	9.49 ± 4.35

Table 2: Evaluation of Alg. 1 for object recognition. Mean and standard deviation error (in %) in estimating the test performance with leave-one-dataset-out. Each row indicates the dataset left out.

formed. For DNNs that overfit less, cavities are formed at smaller ϵ which indicates that fewer connections in the metric spaces are needed to form them.

We show the plots of topological summaries against performance gap for object recognition, AU recognition and

Model	resnet18	vgg16	mean
$g(\lambda)$	6.96 ± 4.18	6.29 ± 3.98	6.62 ± 4.08
$g(\mu)$	4.02 ± 3.51	5.78 ± 3.65	3.83 ± 3.58
$g(\lambda, \mu)$	5.18 ± 3.62	5.87 ± 3.62	5.52 ± 3.62

Table 3: Evaluation of Alg. 1 for AU recognition. Mean and standard deviation error (in %) in estimating the test performance with leave-one-sample-out.

Model	resnet18	vgg16
bp4d	3.82 ± 2.80	6.04 ± 4.17
disfa	3.07 ± 2.17	4.07 ± 3.56
emotionet	7.48 ± 3.66	7.46 ± 5.01

Table 4: Evaluation of Alg. 1 for object recognition. Mean and standard deviation error (in %) in estimating the test performance with leave-one-dataset-out. Each row indicates the dataset left out.

Model	fcn32_vgg16	fcn32_resnet18	mean
$g(\lambda)$	4.86 ± 4.14	6.01 ± 4.34	5.43 ± 4.24
$g(\mu)$	9.46 ± 4.34	4.68 ± 3.82	7.07 ± 4.08
$g(\lambda, \mu)$	5.60 ± 2.86	3.61 ± 3.37	4.60 ± 3.11

Table 5: Evaluation of Alg. 1 in semantic segmentation. Mean and standard deviation error (in %) in estimating the test performance with leave-one-sample-out.

Network	Convolutions	FC layers
conv_2	$64_{3 \times 3} \rightarrow 64_{3 \times 3}$	256, 256, $\#n_classes$
conv_4	$64_{3 \times 3} \rightarrow 64_{3 \times 3} \rightarrow 64_{3 \times 3} \rightarrow 128_{3 \times 3}$	256, 256, $\#n_classes$
conv_6	$64_{3 \times 3} \rightarrow 64_{3 \times 3} \rightarrow 128_{3 \times 3} \rightarrow 128_{3 \times 3} \rightarrow 256_{3 \times 3} \rightarrow 256_{3 \times 3}$	256, 256, $\#n_classes$

Table 6: An overview of the custom deep networks used in this paper. $conv_x$ stands for convolutional networks with x layers. We indicate each convolutional layer by its number of filters followed by their size in subscript and a pooling layer P by its size in subscript. AlexNet [18], VGG [25], and ResNet [15] are standard architectures. Refer to original papers for details.

semantic segmentation in Figures 5-7, respectively. The linear mapping $g(\cdot)$ between each topological summary, life and midlife (Eqs. 3 & 4), and the performance gap are shown in the first and second rows of these figures, respectively. In all these figures rows represent DNN's results.

The results of each dataset are indicated by a disc, where

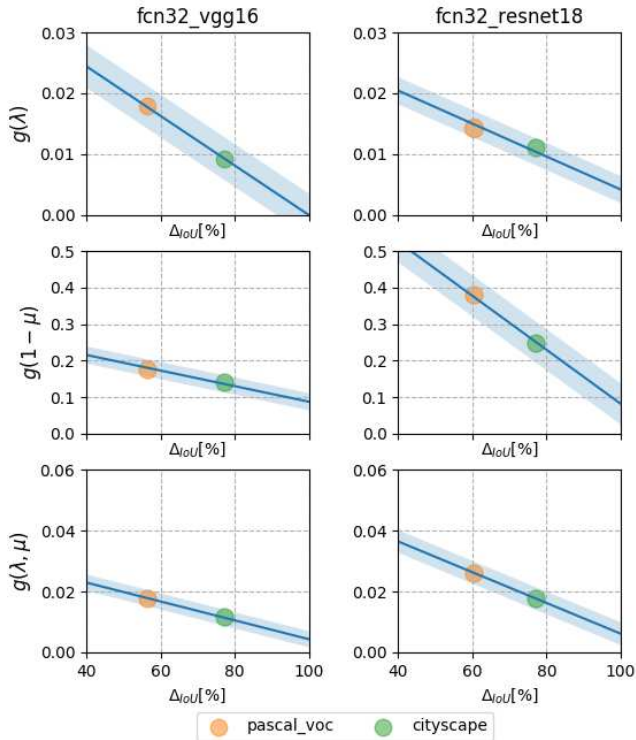


Figure 7: Topology summaries against performance (IoU) gap for different models trained for semantic segmentation. Each disc represents mean (centre) and standard deviation (radius) on a particular dataset. Linear mapping and the corresponding standard deviation of the observed samples are marked.

the centre specifies the mean and the radius the standard deviation. We also mark the linear regression line $g(\cdot)$ and the corresponding standard deviation of the observed samples from it.

Finally, Tables 1, 3 & 5 show the *Error*, namely the absolute value of the difference between the estimate given by Alg. 1 and that of a testing set, computed with Eq. (7) by leaving-one-sample-out. A different way of showing the same results can be found in Tables 2 and 4 where mean and standard deviation of the same error is computed by leaving-one-dataset-out.

It is worth mentioning that our algorithm is general and can be applied to any DNN architecture. In Table 6 we detail the structure of the networks that have been used in this paper. These networks range from simple (with only a few hundred of nodes, *i.e.*, *conv2*), to large (*e.g.*, ResNet) nets with many thousands of nodes.

The strong correlations between basic properties of the functional graph of a DNN and fundamental learning properties like the performance gap also makes these networks more transparent. Not only do we propose an algorithm capable of computing the performance gap, but we show that this is linked to a simple law of the inner-workings of that

networks. We consider this to be a contribution to make deep learning more explainable.

Based on these observations we have chosen to model the relationship between the performance gap and the topological summaries through a linear function. Figures 5-7 show a simplified representation of the observed $(\lambda_i, \mu_i, \Delta_{\rho_i})$ pairs and the regressed lines.

We need to mention that choosing a linear hypothesis for g is by no means the only option. Obviously, using a non-linear regressor for $g(\cdot)$ in Alg. 1 leads to even more accurate predictions of the testing error. However, this improvement comes at the cost of being less flexible when studying less common networks/topologies – overfitting.

Table 1-5 further evaluate the algorithm proposed.

Crucially, an average error between 8.45% and 4.60% is obtained across computer vision problems, which is as accurate as computing a testing error with a labelled dataset \mathcal{Z} .

7. Conclusions

We have derived, to our knowledge, the first algorithm to compute the testing classification accuracy of any DNN-based system in computer vision, without the need for the collection of any testing data.

The main advantages of the proposed evaluation method versus the classical use of a testing dataset are:

1. there is no need for a sequestered dataset to be maintained and updated by a third-party,
2. there is no need to run costly cross-validation analyses,
3. we can modify our DNN without the concern of overfitting to the testing data (because it does not exist), and,
4. we can use all the available data for training the system.

We have provided extensive evaluations of the proposed approach on three classical computer vision problems and shown the efficacy of the derived algorithm.

As a final note, we would like to point out the obvious. When deriving computer vision systems, practitioners would generally want to use all the testing tools at their disposal. The one presented in this paper is one of them, but we should not be limited by it. Where we have access to a sequestered database, we should take advantage of it. In combination, multiple testing approaches should generally lead to better designs.

Acknowledgments. NIH grants R01-DC-014498 and R01-EY-020834, Human Frontier Science Program RGP0036/2016, TIN2016-74946-P (MINECO/FEDER, UE), CERCA (Generalitat de Catalunya) and ICREA (ICREA Academia). CC and AMM defined main ideas and derived algorithms. CC, with SE, ran experiments. CC and AMM wrote the paper.

References

- [1] TinyImagenet Stanford Challenge. <https://tiny-imagenet.herokuapp.com>. Accessed: 2020-03-27. 5
- [2] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018. 5
- [3] L. F. Barrett, R. Adolphs, S. Marsella, A. M. Martinez, and S. D. Pollak. Emotional expressions reconsidered: Challenges to inferring emotion from human facial movements. *Psychological Science in the Public Interest*, 20(1):1–68, 2019. 5
- [4] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017. 5
- [5] C. F. Benitez-Quiroz, R. Srinivasan, Q. Feng, Y. Wang, and A. M. Martinez. Emotionet challenge: Recognition of facial expressions of emotion in the wild. *arXiv preprint arXiv:1703.01210*, 2017. 5
- [6] M. G. Bergomi, P. Frosini, D. Giorgi, and N. Quercioli. Towards a topological–geometrical theory of group equivariant non-expansive operators for data analysis and machine learning. *Nature Machine Intelligence*, 1(9):423–433, 2019. 2, 3
- [7] F. Chazal, D. Cohen-Steiner, L. J. Guibas, F. Mémoli, and S. Y. Oudot. Gromov-hausdorff stable signatures for shapes using persistence. In *Computer Graphics Forum*, volume 28, pages 1393–1403. Wiley Online Library, 2009. 3
- [8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5
- [9] C. A. Corneanu, M. Madadi, S. Escalera, and A. M. Martinez. What does it mean to learn in deep networks? and, how does one detect adversarial attacks? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4757–4766, 2019. 2, 3, 4, 5
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 1, 5
- [11] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 1
- [12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>. 5
- [13] C. Fabian Benitez-Quiroz, R. Srinivasan, and A. M. Martinez. Emotionet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5562–5570, 2016. 1, 5
- [14] A. Hatcher. *Algebraic Topology*. 2002. 2
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6, 7
- [16] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*, 2018. 5
- [17] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2, 6, 7
- [19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015. 1
- [20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 6
- [21] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, and J. F. Cohn. Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, 4(2):151–160, 2013. 5
- [22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5
- [23] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017. 5
- [24] B. Rieck, M. Togninalli, C. Bock, M. Moor, M. Horn, T. Gumbsch, and K. Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. *arXiv preprint arXiv:1812.09764*, 2018. 5
- [25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1, 6, 7
- [26] L. Vietoris. Über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97(1):454–472, 1927. 2
- [27] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015. 2
- [28] A. M. Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019. 2
- [29] X. Zhang, L. Yin, J. F. Cohn, S. Canavan, M. Reale, A. Horowitz, P. Liu, and J. M. Girard. Bp4d-spontaneous: a high-resolution spontaneous 3d dynamic facial expression database. *Image and Vision Computing*, 32(10):692–706, 2014. 5
- [30] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005. 3