Universiteit Leiden

Master Computer Science

Persistent homology in robust image classification

Name:	Antoni Czernek
Student ID:	4000595
Date:	[22/08/2025]
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. Erwin M. Bakker
2nd supervisor:	Prof. Dr. Michael S. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Image classification has been one of the most studied areas in machine learning in recent years. Most of the recently proposed research focuses on data augmentation and improving training algorithms, with adversarial training being a standard method used for improving robustness. On the other hand, we see promising results from a branch of topology that manages to capture geometrical features, difficult to measure using conventional methods. Persistent homology (PH) is one of the most promising tools from Topological Data Analysis that has been applied to image processing. However, the current research in this topic mostly focuses on simple or well-structured data, such as MNIST dataset or medical images. We propose a novel approach to improving image classification robustness that relies on adding topological information using persistent homology and test it on more complex and diverse datasets, using CIFAR-10 and CIFAR-10C, as well as against adversarial attacks. We explore several PH vectorisation methods to determine whether these features can improve robustness without using adversarial training, and which of these methods is the most suitable for image classification tasks. Our findings indicate that adding topological information from persistent homology slightly improves the robustness, compared to a baseline ResNet18 architecture.

1 Introduction

In crucial industries such as medical imaging and autonomous driving, where a wide range of applications of image recognition models exist, it is critical to create models that are stable and cannot be fooled easily. Simply trained deep neural networks are prone to overfit to unreliable patterns, and are shown to be easily fooled even with a single pixel change, as shown by Su et al. [2019]. Discovering such vulnerabilities and implementing methods that tackle them created a new branch of research dedicated to the robustness of neural networks, which is strongly explored to this date.

On the other hand, Topological Data Analysis (TDA) promises to give important geometrical information about the data, which is mostly missed by classical analysis tools. With a strong mathematical background, this approach has seen an increase in applications across various fields, one of them being image classification and processing, where the TDA-based approaches show promising results in improving performance Khramtsova et al. [2022] as well as stability Gabrielsson et al. [2020]. However, so far those methods have primarily focused on simple black and white images, such as handwritten numbers from the MNIST dataset LeCun et al. [2010], or datasets that have similar structures, such as medical imaging data. To our knowledge, TDA has not been applied to more complex datasets like CIFAR-10 or Image-Net. In this thesis, we would like to explore applying Persistent Homology in robust image classification tasks, using a more diverse dataset, CIFAR-10. We will be testing different TDA vectorization approaches to determine if these tools can be successfully applied to difficult datasets, improving performance and/or robustness.

Contributions: This thesis contributes to the research on image robustness classification and persistent homology in three ways:

1. Proposal of two approaches of combining topological features with a widely used image classification network architecture - ResNet18.

2. Evaluation of various persistent homology vectorization methods, and selection of suitable ones for a combination with deep learning networks in image classification tasks.

3. Determination of whether persistent homology can be used to improve image classification robustness, against common corruptions and adversarial attacks, using a more complex dataset than those seen in the literature so far.

2 Related Work

State-of-the-art methods of image classification are shown to be vulnerable to small perturbations in the data Liu et al. [2024]. Developing methods to be robust against those perturbations has become one of the most studied topics in the machine learning research community in recent years.

2.1 Robust classification methods

Assessing models' vulnerabilities often requires a specially created dataset. ImageNet-A and ImageNet-O by Hendrycks et al. [2021b] are a collection of adversarially filtered images, by simply finding images that are related to ImageNet, which the ResNet-50 model misclassifies. Similarly, ImageNet-R Hendrycks et al. [2021a] evaluates whether the model has an abstract knowledge of a class, by testing it on various artistic renditions of the base classes, while ImageNet-C gives a blurred image dataset, giving a real-world scenario to test on.

Overfitting to noise and a lack of generalisation are the most commonly known causes of unstable neural networks. Widely used regularisation methods, such as L1 and L2, dropout, batch normalisation, and early stopping, are employed to mitigate such issues. However, out of the currently available regularisation methods, finding a state-of-the-art, one-size-fits-all approach is not trivial Santos and Papa [2022]. As neural networks rely on their weights, weight decay, also known as L2 Regularisation, emerged as one of the regularisation methods to avoid overfitting. Another approach directly connected to the model's weights, called Adaptive Weight Decay (AWD) Ghiasi et al. [2023] adds a modification to the calculation of the weight decay, to assess accurate regularization and tuning of the hyperparameters.

Data augmentation is a method of increasing the dataset size by using small perturbations, such as color changes, cropping a random part of the images, rotating and mirror-flipping the sample image and adding it back into the training set. These methods have also been shown to improve robustness Rebuffi et al. [2021], as they expose the model to a more diverse sample set, preventing the network from relying on spurious correlations Amerehi

and Healy [2025] and therefore improving generalisation. Additionally, label-mixing methods such as MixUp Zhang et al. [2018] augment the data by adding a combination of samples and their labels into the dataset. This approach encourages a smooth and more linear behaviour between the respective decision boundaries in between training-set examples, creating a more robust system. Similarly, CutMix Yun et al. [2019] inserts a part of one image into another, while creating a new label as a linear interpolation between the combined images. We can also increase the dataset size used for training, by generating additional samples Gowal et al. [2021], as the generative models keep improving, the positive effect on robustness from additionally generated samples in the trainset is also shown to improve. While diversifying training datasets improves robustness, our work explores a complementary approach by enriching features of each individual sample.

We could also expose the network to adversarial examples during training, this improves both robustness and performance, such an approach, also widely known as Adversarial Training (AT) is a powerful tool in various network architectures Zhao et al. [2022], firstly introduced by Goodfellow et al. [2015] AT became a powerful tool to improve robustness Reyes-Amezcua et al. [2024], mostly useful against adversarial robustness Liu et al. [2024]. Currently, a wide range of AT methods are being used, a single-step adversarial training-based approach utilizes a single step gradient to perform the adversarial attack. Since AT comes with significant computational cost, approaches such as Fast Gradient Sign Method (FGSM) Goodfellow et al. [2015] gives an efficient way to generate adversarial examples for training, further developed into Fast Adversarial Training (FAT) methods Pan et al. [2024], Wong et al. [2020], Andriushchenko and Flammarion [2020], Zhao et al. [2023]. To avoid potential `catastrophic overfitting`, caused by the inability to create strong adversarial examples using single step AT, a stronger but more computationally expensive multi-step AT methods were proposed Madry et al. [2018]. While AT achieves robustness by modifying the training process, our approach aims to enrich input features.

2.2 Homology usage

Topological Data Analysis (TDA) emerged as a method to give well-founded geometrical and topological information about the data Chazal and Michel [2021]. A branch of TDA, Persistent Homology (PH) has been introduced as a computational algorithm to output topological information about given data Edelsbrunner et al. [2002]. PH provides novel insights that are difficult to capture using conventional methods Otter et al. [2017]. This novel approach to data analysis quickly found its applications. Identification of breast cancer Nicolau et al. [2011] was one of the first examples that had shown the potential of TDA, since then research community has expanded on the applications of PH in fields such as biology Xia and Wei [2014], Cang and Wei [2018], chemistry Townsend et al. [2020], Murayama et al. [2023], physics Wilding et al. [2021], seismology Sekuloski and Dimitrievska Ristovska [2023], or finance Souto and Moradi [2024]. While the first approaches focused mostly on data analysis with the help of Support Vector Machines, or other shallow learning algorithms. In recent years, researchers started using PH with Deep Learning (DL). In Qiu and Wei [2023], we see a combination of PH with protein language models with fitness predic-

tion for protein engineering problems. Novel topological approaches are particularly useful when analysing graphs; therefore, a trend of implementing PH into Graph Neural Networks (GNNs) appeared, showing improved performance in data sets with strong topological structures Zhao et al. [2020]. Noticeable enhancement can also be seen in graph pooling Ying et al. [2024], where PH helps to integrate global topological invariance to graph pooling layers. On the other hand, various vectorization methods have been proposed to improve the stability of the topological features Ali et al. [2023], these vectorization methods have only recently seen more usage in machine learning with deep neural networks.

2.3 Persistent homology in Image recognition

Structural information, together with resistance to noise Turkeš et al. [2021], created a good input feature for segmentation methods using topological loss as part of the model, to force the neural network to preserve the original topology by utilizing topological loss Stucki et al. [2023]. Similarly, incorporating topological loss in the discriminator of a GAN network enables it to capture more global geometrical structure Bao et al. [2023], giving better results in terms of image generation, together with generating more realistic images. Topological Layer Gabrielsson et al. [2020], which is based on simplicial complex filtrations, is shown to add regularization for machine learning model weights. Additionally, it can be used to perform topological adversarial attacks and incorporate PH into generative models, showing its potential and limitations, especially against cropping a part of an image, as it directly interferes with the underlying topology. While there have been proposed several ways of integrating PH into image classification, such as combining one of the vectorized topological informations - `Persistent Landscapes`, together with a backbone convolutional neural network (CNN) Khramtsova et al. [2022], or using topology embedding and utilizing fully connected blocks to add this information back into the network using matrix multiplication similarly to addition in residual blocks Peng et al. [2024]. To our knowledge, proposed solutions are primarily tested using simple datasets such as MNIST LeCun et al. [2010] or medical images regarding cancer detection, with a lack of research in image classification using PH for more complex datasets. In our work, we aim to extend this line of research to more challenging datasets. In contrast to previous work, we propose an architecture that is more flexible with respect to homology dimension and its vectorization. Furthermore, we explore the use of CNN based residual blocks for feature integration, as an alternative to the linear neural network approaches previously examined.

3 Fundamentals

In this section, we cover the fundamental concepts that are necessary for this thesis. In a later section, we cover the topological background theory in more detail.

3.1 ResNet18

As our "backbone" neural network, we chose ResNet18, introduced in 2016 by He et al. [2016]. This model introduced residual blocks, an architecture that skips connections as shown in Figure 1. The number 18 in the ResNet stands for the number of layers stacked on top of each other, also adding flattening and a fully connected layer, giving us 18 total

layers. Note that in principle, other image classification deep neural network could have been selected as the "backbone" NN, but we opted for ResNet18 as it is widely used in this type of image classification research, and it allows for scaling DNN depth, with models such as ResNet50, or ResNet152.

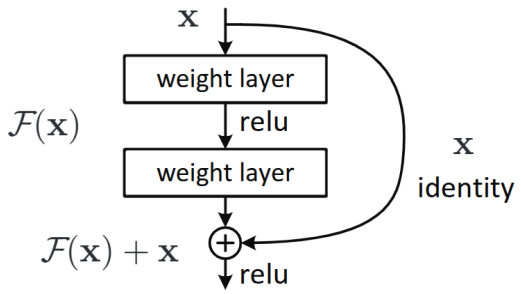

Figure 1: Single residual block architecture

3.2 Data Augmnentation

One of the most powerful tools to tackle robustness is data augmentation, which simply works by creating additional training examples, perturbing the initial data. We will now explain how each perturbation method used in our implementation changes the images. Implemented in the PyTorch library `RandomizedHorizontalFlip` and `RandomVerticalFlip` simply creates additional samples by flipping the original image either horizontally or vertically. As we can see, these methods do not interfere with pixel values, but just rearrange them. Next `RandomRotation` and `RandomPerspective` rearrange the pixels into a different rotation of the original image, or set it into a random perspective. These two methods create black spaces around the image to fill the whole frame. Augmentation that does interfere with pixel values significantly, such as `RandomErasing`, which deletes a small part of the image, `ColorJitter`, which randomly changes the brightness, contrast, saturation and hue, or `GaussianBlur` that blurs the image with a randomly chosen Gaussian blur. Examples of the results of the augmentation methods are depicted in Figure 2.

3.3 Image classification robustness

Small perturbations in data are shown to result in wrong predictions made by the deep-learning models. To measure how our methodology affects model predictions, and therefore how robust our network is with respect to perturbations, we use the common accuracy metric defined as follows:

$$Accuracy(Acc) = \frac{Correct\ Classifications}{Total\ Classifications}$$

Another way to test the robustness is to perform adversarial attacks. To evaluate our models, we use the Auto-Projected Gradient Descent - Cross-Entropy (APGD-CE) attack method, which is a highly effective and widely used algorithm. This approach repeatedly takes small steps in the direction of the gradient of the loss function, trying to maximize the error. The attack is untargeted, meaning that its goal is to simply make the model

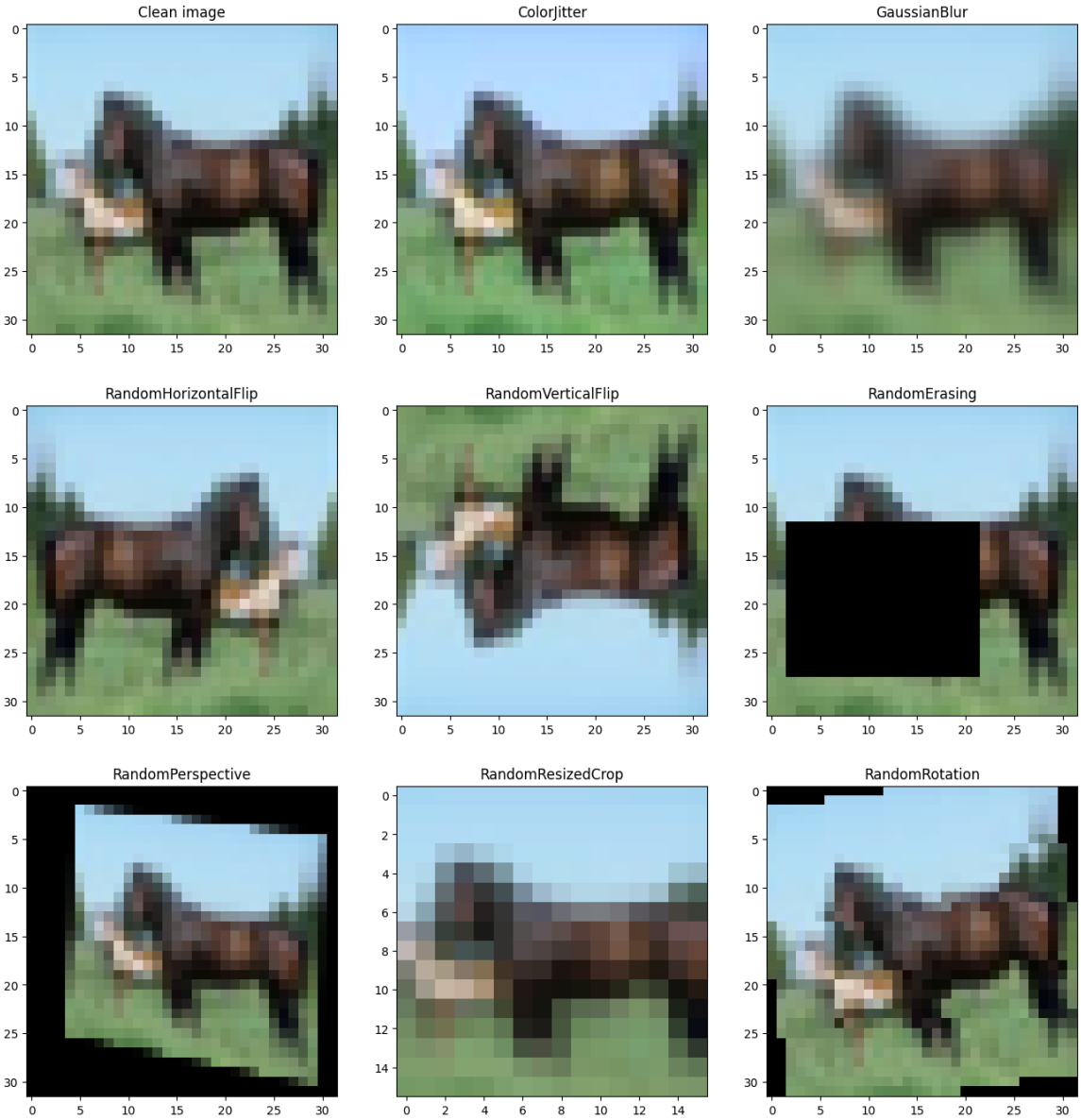

Figure 2: Perturbation methods used for data augmentation.

misclassify the input. We use two norms to quantify the magnitude of the perturbation added to the image, the first one being the L_∞ norm defined as:

$$||x||_\infty = max_i|x_i|$$

This norm simply measures the maximum single pixel difference between a clean sample and a perturbed one. Another norm widely used is the L_2 norm, defined as:

$$||x||_2 = \sqrt{\sum_i x_i^2}$$

For images, this norm measures the Euclidean distance between the original image and the adversarial one. The accuracy in this case is calculated as the number of unsuccessful attacks to the total number of tries.

3.4 Training loss

To train our models, we utilise Cross Entropy Loss. For a single example, it is calculated as:

$$CrossEntropyLoss = -\sum_{i=1}^{C} y_i \log(p_i)$$

Where y_i is an indicator if the i-th class is a correct one, and 0 otherwise, C is the total number of classes, and p_i is the probability given by the model for the i-th class.

4 Topological features

Topological Data Analysis aims to provide additional structural information, mainly focusing on multidimensional "holes" in the vector space constructed from the data. In this section, we will explain how those spaces are constructed, as well as how Persistent Homology is calculated, and what information it provides.

4.1 Persistent Homology

Our goal in PH is to retrieve geometrical and structural information from the data, in particular, finding homology groups in the space created from the data. We will first focus on the procedure for a point cloud, and later show how we can also generalise it to an image.

4.1.1 Creating simplicial complexes

Considering a point cloud in $\mathbb{R}^n$, we want to extract topological features of these points. To do that, we will create a simplicial complex using those points. A Simplicial complex is a topological space created as the sum of simple shapes, called simplices, that are the building blocks for the space. As simplices we consider a point, line, triangle, tetrahedron and other higher-dimensional shapes. Let us also define a simplex wall as the lower-dimensional simplexes contained within the original simplex. For example, for a given 2-dimensional, triangle simplex, the wall will be set as the segments between each point of the original triangle. We illustrate an example of above mentioned definitions in Figure 3, where the walls of the simplex `ABC` will be the set of `AB`, `BC` and `CA`. In our simple example, which is a simplicial complex of the 2-dimensional simplex ABC, which means that we also need to include the vertices `AB`, `BC`, `CA` and the points `A`,`B`, and `C`. Together with that, we also have two 1-dimensional simplexes `AD` and `DC`, which means that we also include the point `D` as part of the simplicial complex.
We create our simplicial complex from a point cloud by putting a ball with radius ϵ in each of the points. We will connect a set of points using our defined building blocks if the respective balls have a non-empty intersection. With such a created simplicial complex, we define a cycle as a set of simplices, that when summing their walls, we count each wall twice. An example of a 1-dimensional cycle would be any loop created from a set of segments. We

also define a border as the walls of a set of simplices. In our example in Figure 3, a border of simplex `ABC` will be the set of segments `AB`, `BC`, `CA`. What we are looking for in such defined spaces are the homology groups, which are generated by the set of cycles that do not belong to the boundary of any set of simplexes. In our example, a cycle $AB \rightarrow BC \rightarrow CA$ is contained in the boundary of `ABC`; therefore, it does not generate a homology group. However, when we examine cycle $AD \rightarrow DC \rightarrow CA$ we cannot find a 2-dimensional set of simplices, whose boundary is the $AD \rightarrow DC \rightarrow CA$ cycle. Therefore, this cycle generates a homology group in one 1-dimension, as it goes around an empty space. We will refer to the dimension of homology groups in each dimension, as the number of holes in that dimension. In this work we only use 0-dimensional and 1-dimensional homology groups, which essentially capture the number of connected components, and number of loops around empty spaces.

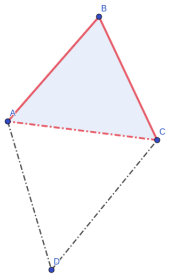

Figure 3: A simple complex with 2-dimensional simplex denoted as `ABC`, and two 1-dimensional simplexes `AD` and `DC`. To satisfy the simplicial complex definition, we also consider the subsimplices of the above-mentioned simplices to also be included in the simplicial complex. A wall of `ABC`, by definition, the 1-dimensional lower subsimplices, are shown as a red line. We can also show the difference between a cycle that generates a homology group and one that is simply a boundary. The red cycle is clearly a boundary of `ABC`, while the cycle $AC \rightarrow CD \rightarrow DA$ shown on the figure as the dashed line generates a homology group, since we cannot find a 2-dimensional set of simplexes whose boundary is the mentioned cycle.

Let us consider a point cloud as depicted in Figure 4, in each of the points in our cloud, we put a ball of radius ϵ, in Figure 4 represented by the orange spheres around the points. Following the previously mentioned method, we will put a simplex between a set of points when the intersection of the respective spheres is non-empty. This is one of the easiest methods to create a simplicial complex, also referred to as Chech Complex. As we increase the ϵ value, we can see that the space is beginning to fill with simplices. Our persistent homology essentially gives us the information of how the homology groups change, with increased ϵ. In this example, we can see that a lot of small holes are created in the process; however, they will quickly vanish when we increase the sphere radius. The one 1-dimensional homology group that will be persistent is the hole in the middle of the circle.

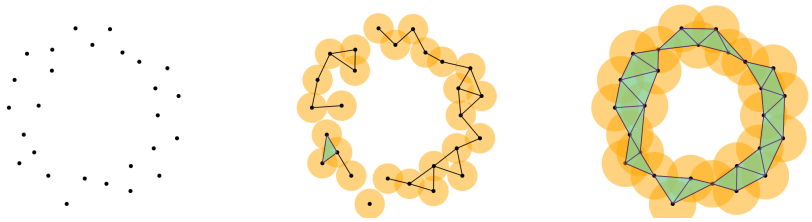

Figure 4: An example of creating a simplicial complex from a point cloud. We first have the data and radius $\epsilon = 0$; therefore, we only see the points. As we increase the radius by increasing the ϵ, denoted in the graph as yellow balls around the points, we see how the space between points is beginning to fill with simplices, creating the simplicial complex space from which we can calculate the homology groups. Visualization provided by Saul [2018]

4.2 Vectorization methods

In order to use the topological features in the neural networks, we first need to vectorize them. We present how the PH can be transformed into a Persistent Diagram, and then we present the vectorization methods used in this work.

4.2.1 Persistent Diagrams

With increased ϵ, the homology groups will appear and disappear; this phenomenon is known as filtration. We can track when a specific homology group has its "birth" and "death", therefore creating a diagram that on the X axis will have the birth of a homology group, and on the Y axis the death of a group. Each point will therefore represent a homology group. We illustrate this on a simple example from the circular data in Figure 5. In this example, when we calculate the homology groups over increased epsilon values, we see that two homology groups will persist the longest, and those are the two holes inside the circles. These holes are represented in the Persistence diagram as two blue points high above the diagonal line. Another key observation to notice is that in this process, small homology groups will appear and quickly disappear, shown by the set of points near the diagonal line on the Persistent diagram. Those homologies are created within the rings of each circle.
Now this diagram can be transformed using various vectorization methods. In this thesis, we utilise a few of those algorithms using the Gudhi Python library The GUDHI Project [2020]. The methods we use are: `Betti curve`, `silhouette`, `landscape`, and `persistent images`. We visualize all of the presented vectorization methods in Figure 5.
To give further context, we give the precise definition of those vectorization algorithms. `Betti Curves` Ali et al. [2023] are created by a simple function

$$\beta_k(t) = Number\ of\ features\ born\ before\ or\ at\ t$$

In other words, this is simply a count of features, present at time t. `Persistent Landscapes` introduced by Bubenik and Dłotko [2017] are calculated using the persistent barcodes, essentially a set of intervals $[b_i, d_i)$ where b_i is the birth of a homology group, and d_i is its

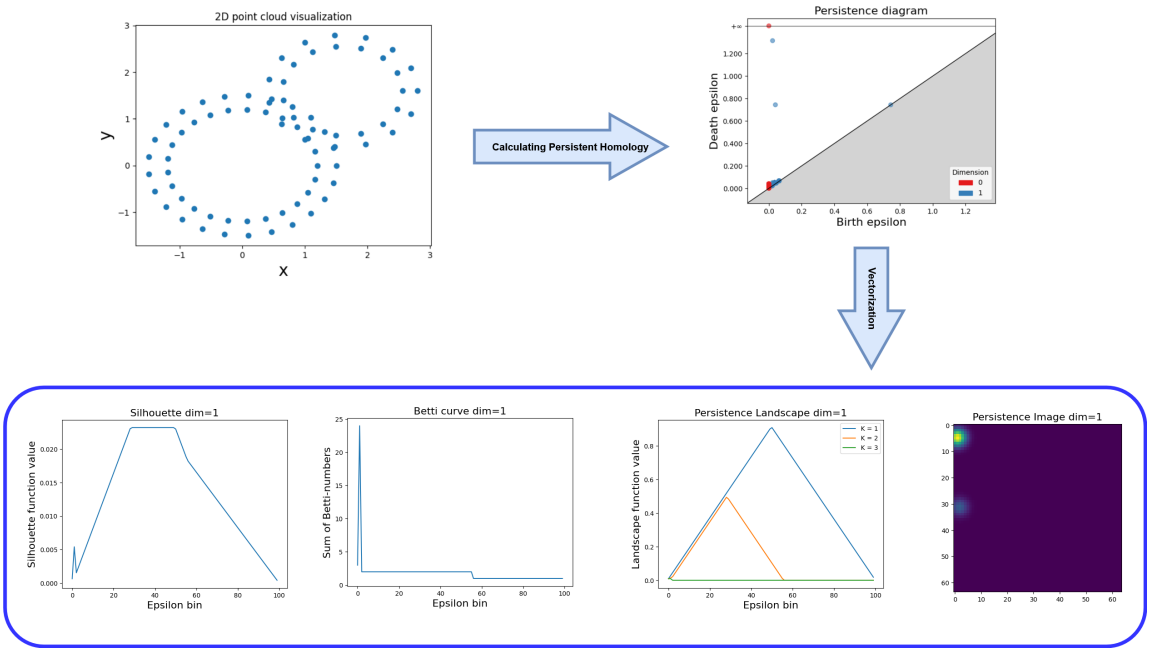

Figure 5: Procedure of generating vectorised topological features from a 2-dimensional point cloud. The two dots in the persistent diagram represent homology groups created by the circles, the lower one for the smaller circle, as it persists shorter, and the higher one for the bigger circle, since it persists longer. The dots near the diagonal line are the small homology groups created between the points on each circle.

death. We then calculate the piecewise linear function $f_{(b,c)} : \mathbb{R} \to [0, \infty]$ defined as follows.

$$f_{(b,d)} = \begin{cases} 0 & \text{if } x \notin (b, d) \\ x - b & \text{if } x \in (b, \frac{b+d}{2}] \\ -x + d & \text{if } x \in (\frac{b+d}{2}, d) \end{cases}$$

A `persistent landscape` is then defined as a sequence of functions $\lambda_k : \mathbb{R} \to [0, \infty)$ for $k \in \mathbb{N}$. The k-th landscape function $\lambda_k(t)$ is defined as the k-th largest value among all the individual functions $f(b, d)$. Now let us define `persistent silhuettes`, we begin with the previously defined function $f(b, d)$, and the key difference from `landscapes` is the weighted average of those tent functions. We define a continuous `silhuette` function $S(t)$ as:

$$S(t) = \frac{\sum_i w_i f_i(t)}{\sum_i w_i}$$

Where $i \in \mathbb{N}$ are the indexes of the homology groups. Since the above mentioned vectorization methods work on continuous values of epsilon, to create a discrete vector that can be used by a DNN we simply sample epsilon values. We refer to these samples as epsilon bins.

The last vectorization method we will look at are persistent images. We begin by transforming the birth and death points into birth and persistence points, by simply subtracting the birth from the death, such that our horizontal line in `persistence diagrams` becomes the

x-axis. We assign weight w to each of the points, which emphasises the points with higher persistence. In each point (b,p) we can center a 2D Gaussian distribution with variance σ^2, therefore our persistence surface is defined as $\phi_D(x,y) = \sum_{(b_i,d_i)\in D} w(b_i,d_i)K_\sigma((x,y) - (b_i,d_i))$. A simple discretisation of this surface results in our `persistent image`. Note that there is no spatial correspondence between the data and the persistent image, the PI reflects only the homology groups. In particular rotating the data while keeping the distances between each point constant, will always result in the same PI.

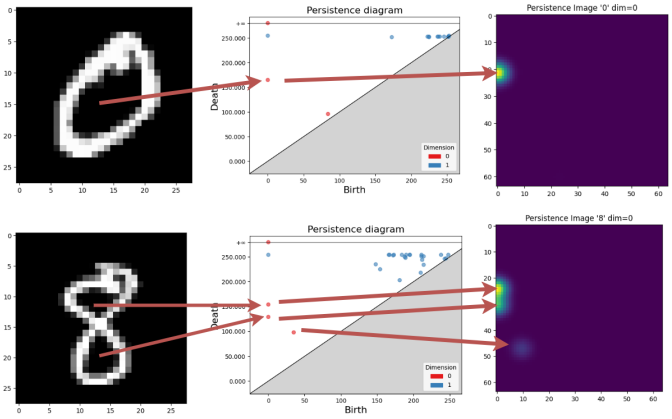

Figure 6: A simple example of how homology groups correspond to features of an image in black and white. We see a single long persisting 0-dimensional homology group for the number "0", and two such groups for an "8". We use red arrows to indicate the features in images that generate the homology groups.

4.2.2 Cubical Complex

So far, we focused on extracting topological features from a point cloud; in order to work with images, we need a different approach to creating simplicial complexes.
In image processing, one of the most widely used TDA methods are cubical complexes, where instead of triangles and tetrahedrons, squares and cubes are used, which naturally translates to images, since each pixel essentially is a square; therefore, each pixel will be a square simplex, together with the vertices and segments between the points in the corners. To calculate the persistent homology of such an image, we need a filtration mechanism. In each step of t we will only consider the simplices, where the pixel value is lower than t, effectively creating a filtration method that can then be used to calculate persistent homology over the image Choe and Ramanna [2022].
An example of such an approach to images is shown in Figure 6, where we demonstrate it on simple images from the MNIST dataset. In this example, the 0-dimensional homology groups - connected components correspond to the number of spaces surrounded by white pixels. In case of a "0", we see one persisting 0-dimensional homology group, which corresponds to the space inside the number "0". On the other hand, for an "8" we see two persisting 0-dimensional groups, one per each space in the "8", also shown by the arrows on the chart. We also see a single long persisting 1-dimensional homology group; this is

simply the white space. Since it's connected in both presented examples, it will create just a single homology group, which is persistent, because the bright pixels will disappear only at the end.

5 Topologically enhanced ResNet

In this section, we will explain the proposed network architectures, topological features vectorization methods, architecture variations, and other details about the proposed approach.

5.1 Topological Residual Network (TRN)

Firstly, we implement a Topological Residual Network (TRN) as depicted in Figure 7, which adds a separate network on top of the plain ResNet model; this additional network calculates the embeddings of the topological features. We embed the output of residual blocks of ResNet-18 into a single vector that is then concatenated with embedded topological features vector, both of these vectors are of size denoted as N_h. The residual embedding network can have 3 sizes: small, which utilises 2 layers and $N_h = 32$, medium size, which consists of 3 layers and $N_h = 64$, and large, which also has 3 layers but with more nodes and an output vector of $N_h = 128$. Residual blocks consist of two convolutional layers. In between, we use batch normalization. After those two convolutional layers, we add the input of the block back into the output of the block, skipping connections.
To calculate the topological features, we first transform the image into a black and white image using OpenCV library Itseez [2015], on which we utilise cubical complexes to calculate the persistent homology, vectorising it and passing it through the topological network. For our vectorization methods, we test this architecture across `persistent landscapes`, `betti curves`, `persistent silhuettes`, and `persistent images` with an image size of 64x64. Another consideration when working with persistent homology is the dimension of homology groups that we want to pass to our network. We either pass, 0-dimensional or 1-dimensional homology groups; we also include a version with concatenated vectors from both dimensions. Vectorization methods that output a single vector are passed through a dense fully connected neural network, called a Topological Embedding Network (TEN). TEN can have three sizes, all of which are 3-layered fully connected linear networks as depicted in Appendix A in Figure 11. This network outputs a vector of the same size as the fully connected network that embeds the residual network output. Input size of TEN is determined by the vectorization methods output vector size.

The two vectors received from the residual network and the topological embedding network, both of size N_h, are then concatenated and passed through the final fully connected network that outputs the prediction. Since the sizes of both residual network embedding and topological network embedding differ by their output vector size, equal to N_h, we test the output NN in 3 sizes, such that we can handle different output vectors. Two of the final network architectures use 3 fully connected layers, while the biggest uses 4. For our activation functions, we use ReLU in each layer except for the final layer, where we use Softmax to output the logits.

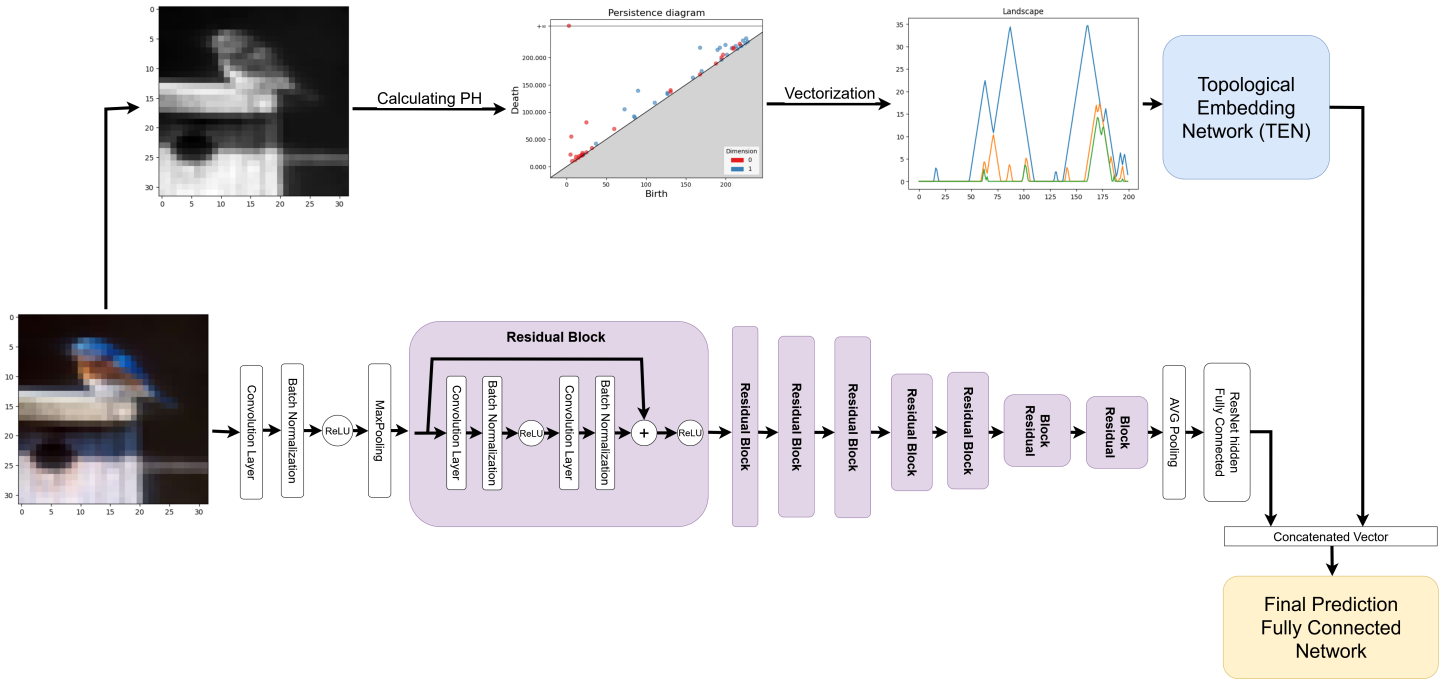

Figure 7: Architecture design for the Topological Residual Network. Topological features are processed separately from the Residual Network, only being concatenated and processed at the end.

5.2 Topological Block Network (TBN)

We also introduce a Topological Block Network (TBN) architecture as depicted in Figure 8, that passes the topological information into the residual blocks. In this implementation, the topological vectorization is passed through a fully connected network `topo-net`, then at each block, when we add the identity vector, we also add the output of `topo-net`. Since input size differs for each block, we introduce a local topology embedding network (LTEN) as depicted in Figure 12 in the Appendix A section. In each block, we create a fully connected network that embeds the previously processed topological vector from `topo-net` and adds to the output of the residual block, before the activation function. Similarly to the `TRN`, we concatenate the output of `topo-net` and the output of the ResNet, process it through a fully connected network to output the class prediction.

5.3 Persistent Images handling

We can pass persistent images into a network in two ways, one as simply a flattened vector. In this case, we can use both previously explained implementations by simply changing the input size of the appropriate networks. We also consider passing topological information in the form of an image of size 64x64 with a single channel. We therefore need to modify TRN and TBN networks to handle those images properly. Those modified architectures are depicted in Appendix B in Figure 13 and 14. For the Topological Residual Block, we use an architecture similar to ResNet to handle the persistent images in a similar manner. That is

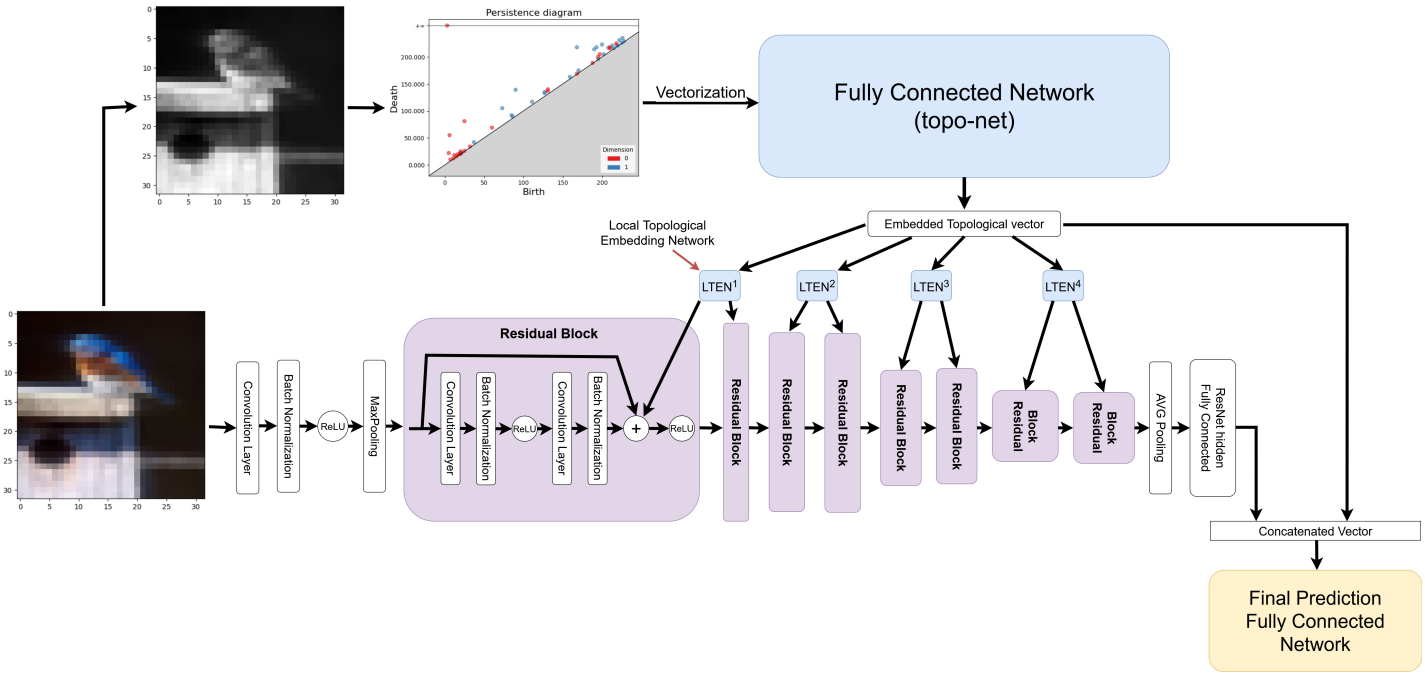

Figure 8: Architecture design for the Topological Block Network. Topological features are embedded and added into the residual blocks, creating `Topological Blocks`.

we pass the original image and the persistent images into two separate networks, concatenate their outputs and pass through a fully connected network to receive the prediction. In case of the TBN, we pass the persistent images through an embedding network to match the ResNet input, then in each block, we pass each of them through a double convolutional layer, separate for the topological information and the base image. Then we add the topological information back into the main ResNet-based network. In the end, we concatenate the outputs similarly to previously mentioned implementations, by concatenating the output of the topological network and image-based network, and passing this vector through a fully connected network.

6 Experimental Setup

To evaluate our implementations, we begin by determining the architecture and hyperparameter setup, then the augmentation methods, and finally evaluate over common corruptions and adversarial attacks.

6.1 Datasets

CIFAR-10 Krizhevsky et al. [2009] is a widely used image classification dataset. It consists of 60,000 32x32 colour images from 10 classes. The dataset is split into 50,000 training examples, which we split into 40,000 images our networks are trained on and 10,000 validation examples. This split is the same across all performed training runs. The 10,000 test images provided by CIFAR-10 are only used in the final clean evaluation, as well as the evaluation

over adversarial attacks. We also utilize the CIFAR-10C dataset Dan Hendrycks [2019], which is a variation of the original CIFAR-10, with common corruptions, such as gaussian noise, a set of few different blur effects, and more. This dataset will be used to determine the robustness of our implementations.

6.2 Architecture and hyperparameter evaluation

Firstly, we determine the architectures and augmentation sets suitable for further testing. We train each combination of architecture, persistent homology vectorization method, and homology dimension setting over three learning rates: $lr = 0.001$, $lr = 0.0003$, or $lr = 0.00001$. Overall, in this first set of runs, we test nearly 400 different configurations. The models were not initialized with pre-trained weights but trained from scratch on the target dataset.
We then select the most promising implementations for each vectorization method and retrain them, with using additional samples from a set of different augmentation sets. We split the augmentation methods into three sets, depending on their interference with topological features of the underlying images: a set that does not interfere with the topology of the images in a significant way. This set of augmentation methods consists of `RandomHorizontalFlip`, `RandomVerticalFlip`, `ColorJitter`, and `GaussianBlur`. Since they mostly re-arange the pixels, or change the underlying values slightly, they only slightly interfere with the persistent homology of the images. Another set of augmentation methods is the set that significantly interferes with the underlying topology; this one consists of `RandomErasing`, `RandomResizedCrop`, `RandomRotation` and `Random-Perspective`. These methods significantly change the pixel values or create black spaces, which change the PH of the images. We also combine those two sets to create the `all` augmentation method. Selected models are trained using a single set of augmentation methods mentioned above, which adds either 30% or 50% additional training samples. We run around 170 different models and augmentation combinations to determine the ones used for the final training and evaluation. The selection is based on the validation accuracy, where we chose the best performing method, ensuring at least one model for each vectorization method. The selected models are shown in Table 2.

6.3 Robustness evaluation

Once we determine the architectures, hyperparameters values, and augmentation details, we evaluate the clean and robustness performance of the proposed implementations. We begin by using the original test set from CIFAR-10 to retrieve the clean accuracy. Then, we use CIFAR-10C for the evaluation of the robustness over common corruptions. Finally, we test our methods over the adversarial attacks, using the Auto-Projected Gradient Descent - Cross-Entropy (APGD-CE) method with norms $L_\infty = \frac{8}{255}$ and $L_2 = 0.5$, since these values are used for the official RobustBench Croce et al. [2021] evaluation benchmarks. The implementation of mentioned adversarial attacks is provided by this framework as well.

6.4 Model wrapping

Provided datasets and methods from the RobustBench library are using already standardized values for testing. Since our implementations calculate persistent homology over the clean 0-255 valued samples, we need to add additional functionality to firstly retrieve the image in appropriate values. To achieve that we simply multiply the data by 255 and perform similar steps as in the train and validation sets preparation. We transform the images into black and white images using OpenCV Itseez [2015] and calculate persistent homology using Gudhi library The GUDHI Project [2020]. Finally, we normalize the values by subtracting the minimum and dividing by the difference between the maximum and the minimum. The maximum and minimum values are taken only from the training set.

6.5 Current state of the art comparison setup

We compare our implementations on the corruption CIFAR-10C dataset and against adversarial attacks, using current best performing methods, which use ResNet18 or similar architectures and are benchmarked on CIFAR-10 dataset on the RobustBench leaderboard Croce et al. [2021].
First approach we compare to is `DAJAT` introduced by Addepalli et al. [2022]. This method combines the AutoAugment method Cubuk et al. [2019], which aims to improve generalisation capabilities, by finding the best augmentation policies to be used for training. The authors combine AutoAugment with traditional clean training by using the same weights for both of scenarios, but with separated batch normalizations. The loss used to train this approach is the combination of augmented and clean runs, using the Jensen-Shannon divergence loss. Finally, adversarial training is used to further improve the robustness by maximizing the KL divergence between the predictions of clean and adversarial samples; furthermore, the authors also use Adversarial Weight Perturbation for training.
As the second approach, we compare to the work proposed by Sehwag et al. [2021], which shows that using proxy distributions from Generative Adversarial Models or Diffusion-based models can improve the robustness. The authors propose a few methods of adding the hidden distributions of those generative models for additional data augmentation. In this approach, the training is also performed with the use of adversarial training using projected gradient descent-based adversarial examples. We refer to this model as `ProxyDist` in the results table.
An algorithm created by Rade and Moosavi-Dezfooli [2021] is another method we will compare our results to. This algorithm aims to reduce the trade-off between clean accuracy and robust accuracy by adding wrongly labelled examples to the training set. The authors mimic the features learned by a clean training sample set by incorporating a set of `helping examples`. The proposed approach, called `Helper-based Adversarial Training` or `HAT`, also shows significant improvement in robustness. Not that all methods we compare to are using some form of adversarial training.

7 Experimental results

We present the results of the evaluation on clean and corrupted examples, and finally against adversarial attacks. Next, to assess topological feature importance, we visualise a part of the trained TRN network weights from a single layer.

7.1 Architecture and hyperparameter evaluation results

Firstly, we determine the architecture and hyperparameters of the underlying models, as well as the augmentation setting on which we retrieve the best performance

Aggregated implementation	Min Acc (%)	Max Acc (%)	AVG Acc (%)	STD (%)
TRN	**61.15**	83.93	76.28	5.18
TBN	61.00	**84.29**	**76.63**	5.18
Persistent Images (Image)	**75.75**	**84.29**	**80.94**	2.13
Persistent Images (Vector)	64.89	83.47	75.45	4.94
Landscape	61.15	83.93	75.88	5.83
Betti Curves	67.44	83.51	76.28	4.40
Silhuette	61.18	83.85	75.44	5.13
0 dimensional homology	61.15	**84.29**	76.36	5.11
1 dimensional homology	**62.97**	83.85	76.20	5.12
concatenated 1 and 0	61.00	83.79	**77.33**	5.40

Table 1: Aggregated results for the first set of runs using clean dataset for training. Presented accuracy is the validation accuracy, aggregated over all runs that use a specific implementation detail.

7.1.1 Architecture design evaluation

The results for the initial training runs, on a clean dataset are summarised in Table 1. Overall, we see no significant difference between TRN and TBN implementations. Dimensionality also does not seem to be a significant contributor to the final results. The biggest influence over the results seems to be the vectorisation method, with Persistent Images in the Image form outperforming the rest. We therefore assume that the most significant influence over the results comes from the combination of the underlying specific network architectures, such as small, medium or large embedding networks, and hyperparameters such as learning rate. In particular we noticed that the lowest learning rate setting would result in better performance. For further evaluation, we chose the best performing models based on validation accuracy. We also make sure that each vectorization method and network architecture are present in the next evaluations.

7.1.2 Augmentation settings evaluation

Experiments on the augmentation settings were conducted using the best-performing models selected from the initial setups, based on their validation accuracy. The results for the types of augmentation as well as the percentage of augmentation used are summarised in the Appendix C section in Table 6. We notice a significant difference in the accuracy when

Model Name	PH vectorization	PH dimension	Augmentation type
ResNet18	-	-	non-topo
BC-TBN-0	Betti-Curve	0	all
BC-TBN-concat	Betti-Curve	0 and 1	all
Land-TBN-concat	Landscape	0 and 1	all
Land-TBN-1	Landscape	1	all
Silhuette-TBN-0	Silhuette	0	all
Silhuette-TBN-concat	Silhuette	0 and 1	all
PI-IMG-TBN-concat	Persistent Image (Image)	0 and 1	all
PI-IMG-TRN-concat	Persistent Image (Image)	0 and 1	non-topo

Table 2: Selected implementations for the final evaluation section, together with their persistent homology vectorization method, homology dimension and type of the augmentation used for training.

Model Name	Val Acc (%)	Clean Test Acc (%)	Cifar-10C Acc (%)
ResNet18 - with augmentation	**92.24**	**86.20**	51.78
BC-TBN-0	88.34	82.70	49.19
BC-TBN-concat	88.16	82.70	48.82
Land-TBN-concat	91.01	82.8	49.51
Land-TBN-1	90.73	84.8	**54.17**
Silhuette-TBN-0	89.31	81.9	48.77
Silhuette-TBN-concat	88.88	81.6	49.08
PI-IMG-TBN-concat	91.47	84.1	52.49
PI-IMG-TRN-concat	90.74	84.9	53.44

Table 3: Overall results of the best performing proposed methods, on the validation set, clean test data and common corruption dataset.

selecting 50% or 30% additional augmented samples, with 89.03% average accuracy and 86.34% respectively. Across all metrics, the higher number of examples added to the train set improves the performance. However, we also observe that the type of augmentation seems to be an insignificant contributor to the final results. The `Topological` augmentation type does achieve the highest average of 87.84% accuracy, while the `All` type does provide higher maximum results. We noticed that in the top results, a single architecture performs similarly across all augmentation types.
Therefore, all selected models used for the final evaluations are using the 50% additional samples from the augmentation. Architectures of the models used for the robustness evaluations are shown in Table 2.

7.2 Robusntess evaluation results

We evaluate our proposed methods, on the clean test set from CIFAR-10, as well as on common corruptions dataset CIFAR-10C. Finally we perform adversarial attacks against our methods, to further evaluate the robustness.

7.2.1 Clean and Common Corruption tests

The results from the original test set from CIFAR-10 as well as the common corruptions dataset CIFAR-10C are shown in Table 3. The detailed results from each corruption type are shown in the Appendix D section in Table 7.
As a baseline for this evaluation, we are using the ResNet18 implementation, which uses 50% augmented training data. On the clean test, we notice that none of our proposed methods match the result of the baseline ResNet. Highest clean test accuracy for our models is achieved by `PI-IMG-TRN-concat` version, which classified the test samples with 84.9% accuracy, compared to the baseline result of 86.2%.
On the common corruption set, the majority of the models using a vector topological input underperform the baseline ResNet18. Out of these methods, only a single model that utilises `Landscape` vectorisation manages to perform better in an overall corruption accuracy over the ResNet. Since `Persistent Landscapes` were created as a stable and robust vectorisation of the homological information, we did expect those models to perform better on robustness evaluation. This might suggest that this approach is not very suitable for image processing. Both models, based on the persistent images, however, do outperform the baseline model, one using TRN and the other using TBN architecture, with concatenated information from both of the homology groups dimensions. Persistent images are known for their stability and provide detailed information about the topology. In contrast to `Landscape` vectorisation, this method shows more promising results across the validation set, clean set and common corruption set.
Another key observation we noticed is that each of the presented models outperforms the baseline ResNet model on specific types of corruptions. In particular, on corruption types: `elastic transform`, `fog`, `glass blur`, `impulse noise`, `JPEG compression`, and `pixelate`, all topologically based models outperform the baseline, in some cases by up to 63% on the `impulse noise` corruption method. However, we also do note that as we see improvement in some corruption types, we also see that the topological models struggle with other corruption types. In particular, on `gaussian noise` corruption type, we see a drop in performance by up to 41%.

7.2.2 Adversarial attack robustness

We test the models robustness against adversarial attacks, by using the APGD-CE untargeted attack method Croce and Hein [2020] with $L_\infty = 8/255$ and with $L_2 = 0.5$, similarly to the values presented in the leaderboard of RobustBenchmark Croce et al. [2021]. Results of these runs are shown in Table 4. Furthermore, we run the tests over increasing epsilon values for both norms; these results are shown in Figure 9.
These attack methods successfully fool the ResNet18 trained with augmentation on each example, quickly dropping the accuracy to 0% in the L_∞ method, when we increase epsilon values. While using the L_2 norm, we also notice a significant and rapid drop in performance, with the final result over epsilon equal to 0.5 resulting in 0.2% accuracy. Each of the tested implementations that use persistent homology outperforms ResNet results in all configurations.
Highest results coming from two methods, which previously performed poorly on clean CIFAR-10 and CIFAR-10C, like `BC-TBN-0` and `Silhuette-TBN-0`. However, the previously

Model Name	Linf = 8/255 Acc (%)	L2 = 0.5 Acc (%)
ResNet18	0.00	0.20
BC-TBN-0	**12.0**	18.9
BC-TBN-concat	7.0	11.8
Land-TBN-concat	1.6	6.3
Land-TBN-1	1.5	5.7
Silhuette-TBN-0	8.7	**19.4**
Silhuette-TBN-concat	6.1	10.7
PI-IMG-TBN-concat	7.6	14.8
PI-IMG-TRN-concat	3.7	13.0

Table 4: Results from the adversarial attack evaluation, using APGD-CE method, with $L_\infty = 8/255$ norm and $L_2 = 0.5$ norm as the limit for the perturbation.

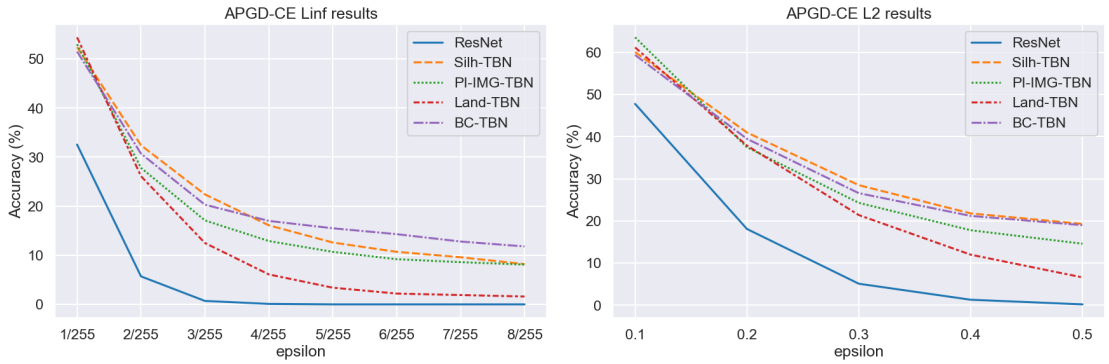

Figure 9: Accuracy against adversarial attacks, with increasing epsilon values.

well-performing landscape-based methods now perform the worst compared to other topological models. Increasing epsilon values comes with a significant drop in performance, similar to ResNet results. However, in this case, the accuracy does not drop to 0 even with the highest tested epsilon setting.
Note that even though persistent image methods do not achieve the highest results, they still give a solid performance, making them a stable and reliable PH-based implementation across all benchmarks.

7.3 Current state of the art comparison results

To further evaluate our proposed method, we compare it to current state of the art methods, that also utilize ResNet18 as the base model, and are currently the top performing approaches on RobustBench leaderboard Croce et al. [2021] among ResNet18 based architectures. For comparison methods, we chose the `DJAT` Addepalli et al. [2022], `ProxyDist` Sehwag et al. [2021] and `HAT` Rade and Moosavi-Dezfooli [2021]. Out of all methods we proposed in this work, we chose the persistent images based model, since it performed well across all previous evaluation tests.
The results are shown in Table 5, where we compare the accuracy on clean test, the CIFAR-

Model	Clean (%)	CIFAR-10C (%)	Linf (%)	L2 (%)
DJAT Addepalli et al. [2022]	86.6	71.2	56.0	69.1
ProxyDist Sehwag et al. [2021]	85.2	69.5	58.9	66.0
HAT Rade and Moosavi-Dezfooli [2021]	86.7	69.4	61.0	68.4
PI-IMG-TBN-concat	84.1	52.5	7.6	14.8

Table 5: Accuracy comparison of our most prominent model and state-of-the-art methods, on the clean test set, common corruptions test set and against APGD-CE adversarial attacks using Linf and L2 norms. All of the methods we are comparing to are using adversarial training as part of the implementation, and are based on the ResNet18 architecture.

10C dataset, and adversarial attacks using $L_\infty = 8/255$ and $L_2 = 0.5$ norms for APGD-CE adversarial attack method. Our approaches give significantly lower performance across both robustness tests. We note that each method we compare to uses some form of adversarial training, and therefore, is exposed to such samples during training. A possible conclusion would be that while persistent homology does give additional information that helps a base network against adversarial attacks, it is not enough to reduce their effect significantly, while training on adversarial samples gives a strong advantage against such attacks. Lower accuracy on the CIFAR-10C dataset and clean testset, also showcases the strength of adversarial training, compared to our proposed method.

7.4 Topological information in network weights

As the last part of the experiments, we aimed to determine whether the topological information contributes to the final prediction.

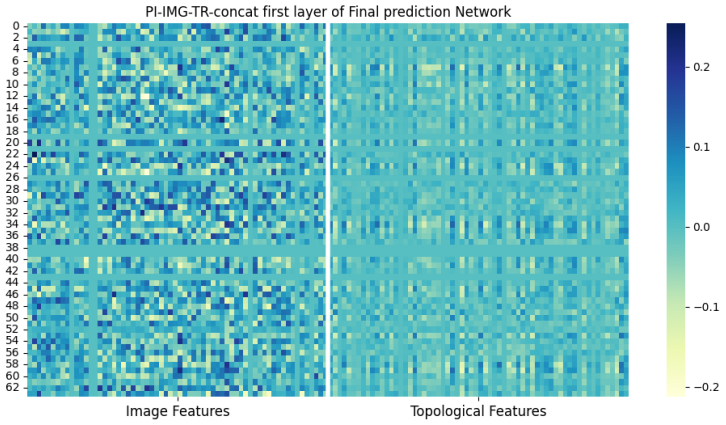

Figure 10: Visualisation of the first layer of the final fully connected network of the PI-IMG TRN-based architecture. Since the embedded topological features and image features are concatenated before this layer, it is a suitable place to check whether the topological features are being utilised by the network.

We took the only TRN model from our finally selected implementations and visualised the first layer of the last fully connected network. The TRN network is necessary to determine the feature importance, since the topological information is processed separately from the image features. These two processed features are concatenated and processed by the final prediction network. We visualize the first layer of this network in Figure 10.
Since this matrix of weights works on the concatenation, we know that image features are processed by the 0-63 columns, and the topological features are processed by the 64-127 columns, in this particular network architecture.
Out of this heatmap, we note that the topological features seem to slightly contribute to the final prediction. However, their importance is significantly lower than the image features. We draw this conclusion from the observation that the image part of the weights shows a lot of different and high value variations, while the topological side gives lower and less spread values, with some of the features being close to 0. This indicates that the network does not utilize the topological features as much as the image features. Since the homological information was an add-on to the base image information, this result adds additional context, which might explain the lower clean performance of the topological models and presents a possible shortcoming for such approach image classification methods with topological features.

8 Conclusions

From the above-mentioned experiments, we arrived at the conclusions about the proposed methodology for approaching image classification robustness using persistent homology.

8.1 Overall conclusion

We propose a method of tackling robust image classification by adding topological features using persistent homology. We test our implementations on more complex data, compared to previously studied literature. Using a wide range of experiments, we show that the architecture design of how the topological features are added into the model does not influence the clean performance greatly; however, TBN architecture with interactions between topological features and image features seems to perform slightly better, which resulted in this particular implementation being primarily chosen for the final evaluations. Primarly persistent image based TBN network, which performed well across all benchmarks.
Out of the tested topological vectorisation methods, `persistent images` show promising results across all benchmarks, giving stable results in all runs, being among the top-performing methods and outperforming baseline ResNet in all of the robustness tests. Other vectorised methods, such as `landscape`, `Silhuette`, and `Betti curve` perform well either at clean samples, corrupted samples or against adversarial attacks, but struggle to perform well in all of the tests.
We do note that these features seem to be slightly under-utilised by the tested persistent image-based TRN network. Since the images provide more information regarding the label, PH is an addition that should give a better idea of the concept of a specific class, which is more robust against small image perturbations. Overall, we arrive at the conclusion that persistent homology methods can be used to improve the robustness of image classification

models; however, adding topological features in a concise does not provide a significant performance boost.

8.2 Contributions

We contribute to the field of image classification robustness and topological data analysis by the following contributions:

1. Proposal of two approaches - TRN, TBN, together with their PI variations, for combining topological features with a widely used image classification network architecture - ResNet18.

2. Evaluation of `landscape`, `silhouette`, `BettiCurve` and **persistent images** vectorization methods for persistent homology, and selection of suitable ones for a combination with deep learning networks in image classification tasks.

3. Our research shows that persistent homology slightly improves image classification robustness for more complex and diverse datasets (CIFAR-10, CIFAR-10C). Against common corruptions by up to 4.5% accuracy improvement, and by up to 19.2 pp improvement against adversarial attacks.

8.3 Future work

As future work, we suggest training the networks with topological features, using adversarial training techniques. Since the addition of persistent homology improves the robustness slightly when compared to the baseline ResNet18 model, we expect the adversarial attacks to further enhance the robustness of the presented implementations. We also note that all currently most promising methods we compare use adversarial training and result in significantly better overall performance.
Furthermore, while CIFAR-10 is significantly more complex than previously studied MNIST or medical images, it still possesses some limitations. To address this, further testing on larger images and even more complex datasets, such as CIFAR-100 or Image-Net, could give a better overview of how topological features influence the robustness.
The slight under-utilisation of the topological features may be the main cause of the minor performance increase; therefore, a further loss function could be used to encourage the network to utilise those features more, possibly improving the robustness further.

References

Sravanti Addepalli, Samyak Jain, et al. Efficient and effective augmentation strategy for adversarial training. *Advances in Neural Information Processing Systems*, 35:1488–1501, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/09d22e4155aa4fdadf3dac8c6bd940fe-Paper-Conference.pdf.

Dashti Ali, Aras Asaad, Maria-Jose Jimenez, Vidit Nanda, Eduardo Paluzo-Hidalgo, and Manuel Soriano-Trigueros. A survey of vectorization methods in topological data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14069–14080, 2023. URL http://dx.doi.org/10.1109/TPAMI.2023.3308391.

Fatemeh Amerehi and Patrick Healy. Narrowing class-wise robustness gaps in adversarial training. *The International Conference on Learning Representations (ICLR) 2025 Workshop on Foundation Models in the Wild*, 2025. URL https://openreview.net/pdf?id=23zngyObBI.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b8ce47761ed7b3b6f48b583350b7f9e4-Paper.pdf.

Jinri Bao, Zicong Wang, Junli Wang, and Chungang Yan. Persistent homology based generative adversarial network. In *VISIGRAPP (4: VISAPP)*, pages 196–203, 2023.

Peter Bubenik and Paweł Dłotko. A persistence landscapes toolbox for topological statistics. *Journal of Symbolic Computation*, 78:91–114, 2017. URL http://dx.doi.org/10.1016/j.jsc.2016.03.009.

Zixuan Cang and Guo-Wei Wei. Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction. *International Journal for Numerical Methods in Biomedical Engineering*, 34(2):e2914, 2018. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.2914.

Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists. *Frontiers in artificial intelligence*, 4 - 2021, 2021. URL https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2021.667963/full.

Seungho Choe and Sheela Ramanna. Cubical homology-based machine learning: An application in image classification. *Axioms*, 11(3), 2022. ISSN 2075-1680. doi: 10.3390/axioms11030112. URL https://www.mdpi.com/2075-1680/11/3/112.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, volume 119, pages 2206–2216. PMLR, 2020. URL https://proceedings.mlr.press/v119/croce20b.html.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *NeurIPS 2021 Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=SSKZPJCt7B.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Cubuk_AutoAugment_Learning_Augmentation_Strategies_From_Data_CVPR_2019_paper.html.

Thomas Dietterich Dan Hendrycks. Benchmarking neural network robustness to common corruptions and perturbations. *The International Conference on Learning Representations (ICLR) 2019*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm&hl=es.

Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002. URL https://link.springer.com/article/10.1007/s00454-002-2885-2.

Rickard Brüel Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020. URL https://proceedings.mlr.press/v108/gabrielsson20a.html.

Mohammad Amin Ghiasi, Ali Shafahi, and Reza Ardekani. Improving robustness with adaptive weight decay. *Advances in Neural Information Processing Systems*, 36:79067–79080, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f9d7d6c695bc983fcfb5b70a5fbdfd2f-Paper-Conference.pdf.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *The International Conference on Learning Representations (ICLR) 2015*, 2015. URL https://arxiv.org/abs/1412.6572.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. *Advances in neural information processing systems*, 34:4218–4233, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/21ca6d0cf2f25c4dbb35d8dc0b679c3f-Paper.pdf.

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, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International*

Conference on Computer Vision (ICCV), pages 8340–8349, October 2021a. URL https://openaccess.thecvf.com/content/ICCV2021/html/Hendrycks_The_Many_Faces_of_Robustness_A_Critical_Analysis_of_Out-of-Distribution_ICCV_2021_paper.html.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15262–15271, 2021b. URL https://openaccess.thecvf.com/content/CVPR2021/html/Hendrycks_Natural_Adversarial_Examples_CVPR_2021_paper.html.

Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015. Accessed in February 2025.

Ekaterina Khramtsova, Guido Zuccon, Xi Wang, and Mahsa Baktashmotlagh. Rethinking persistent homology for visual recognition. In *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, 2022. URL https://proceedings.mlr.press/v196/khramtsova22a.html.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. MS Thesis, University of Toronto.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Chang Liu, Yinpeng Dong, Wenzhao Xiang, Xiao Yang, Hang Su, Jun Zhu, Yuefeng Chen, Yuan He, Hui Xue, and Shibao Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. *Int. J. Comput. Vision*, 133(2): 567–589, 2024. URL https://doi.org/10.1007/s11263-024-02196-3.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *The International Conference on Learning Representations (ICLR) 2018*, 2018. URL https://openreview.net/pdf?id=rJzIBfZAb.

Burai Murayama, Masato Kobayashi, Masamitsu Aoki, Suguru Ishibashi, Takuya Saito, Takenobu Nakamura, Hiroshi Teramoto, and Tetsuya Taketsugu. Characterizing reaction route map of realistic molecular reactions based on weight rank clique filtration of persistent homology. *Journal of Chemical Theory and Computation*, 19, 2023. URL https://doi.org/10.1021/acs.jctc.2c01204.

Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011. URL https://www.pnas.org/doi/abs/10.1073/pnas.1102826108.

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(17), 2017. URL http://dx.doi.org/10.1140/epjds/s13688-017-0109-5.

Chao Pan, Qing Li, and Xin Yao. Adversarial initialization with universal adversarial perturbation: A new approach to fast adversarial training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21501–21509, 2024. URL https://ojs.aaai.org/index.php/AAAI/article/view/30147.

Yaopeng Peng, Hongxiao Wang, Milan Sonka, and Danny Z Chen. Phg-net: Persistent homology guided medical image classification. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7583–7592, 2024.

Yuchi Qiu and Guo-Wei Wei. Artificial intelligence-aided protein engineering: from topological data analysis to deep protein language models. *Briefings in bioinformatics*, 24(5): bbad289, 2023.

Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Data augmentation can improve robustness. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, page 29935–29948, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 978-1-7138-4539-3.

Ivan Reyes-Amezcua, Gilberto Ochoa-Ruiz, and Andres Mendez-Vazquez. Enhancing image classification robustness through adversarial sampling with delta data augmentation (dda). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 274–283, June 2024.

Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Computing Surveys*, 54(10s):1–25, January 2022. ISSN 1557-7341. doi: 10.1145/3510413. URL http://dx.doi.org/10.1145/3510413.

Nathaniel Saul. Chech complex playground visualisation website. https://sauln.github.io/blog/nerve-playground, 2018. Accessed in February 2025.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? *The International Conference on Learning Representations (ICLR) 2022 Poster*, 2021. URL https://openreview.net/forum?id=WVXONNVBBkV.

Petar Sekuloski and Vesna Dimitrievska Ristovska. Application of persistent homology in tda – a case study on seismology dataset. *ICT Innovations 2023, Web Proceedings ISSN null*, 2023.

Hugo Gobato Souto and Amir Moradi. A novel loss function for neural network models exploring stock realized volatility using wasserstein distance. *Decision Analytics Journal*,

10:100369, 2024. ISSN 2772-6622. doi: https://doi.org/10.1016/j.dajour.2023.100369. URL https://www.sciencedirect.com/science/article/pii/S2772662223002096.

Nico Stucki, Johannes C. Paetzold, Suprosanna Shit, Bjoern Menze, and Ulrich Bauer. Topologically faithful image segmentation via induced matching of persistence barcodes. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023. URL https://proceedings.mlr.press/v202/stucki23a.html.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.1.1 edition, 2020. URL https://gudhi.inria.fr/doc/3.1.1/. Accessed in May 2025.

Jacob Townsend, Cassie Putman Micucci, John H Hymel, Vasileios Maroulas, and Konstantinos D Vogiatzis. Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nature communications*, 2020.

Renata Turkeš, Jannes Nys, Tim Verdonck, and Steven Latré. Noise robustness of persistent homology on greyscale images, across filtrations and signatures. *Plos one*, 16(9):e0257215, 2021.

Georg Wilding, Keimpe Nevenzeel, Rien van de Weygaert, Gert Vegter, Pratyush Pranav, Bernard JT Jones, Konstantinos Efstathiou, and Job Feldbrugge. Persistent homology of the cosmic web–i. hierarchical topology in λcdm cosmologies. *Monthly Notices of the Royal Astronomical Society*, 2021.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *The International Conference on Learning Representations (ICLR) 2020*, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

Kelin Xia and Guo-Wei Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International journal for numerical methods in biomedical engineering*, 30 (8):814–844, 2014.

Chaolong Ying, Xinjian Zhao, and Tianshu Yu. Boosting graph pooling with persistent homology. In *Advances in Neural Information Processing Systems*, volume 37, page 19087–19113. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/21f76686538a5f06dc431efea5f475f5-Paper-Conference.pdf.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR 2018*, 2018. URL https://openreview.net/pdf?id=r1Ddp1-Rb.

Mengnan Zhao, Lihe Zhang, Yuqiu Kong, and Baocai Yin. Fast adversarial training with smooth convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4720–4729, October 2023.

Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, page 2896–2906. PMLR, 2020. URL https://proceedings.mlr.press/v108/zhao20d.html.

Weimin Zhao, Sanaa Alwidian, and Qusay H. Mahmoud. Adversarial training methods for deep learning: A systematic review. *Algorithms*, 15(8), 2022. ISSN 1999-4893. doi: 10.3390/a15080283. URL https://www.mdpi.com/1999-4893/15/8/283.

Appendix A. Fully connected topological network architectures

Detailed architectures of the Topological Embedding Network (TEN) of small size is depicted in Figure 11. Where N_t is the size of the vectorized persistent homology, and N_h is the hidden vector size. Note that sizes to which we refer to as medium and large differ only by the number of neurons in each layer, $(N_t \times 128)$, (128×128), $(128 \times N_h)$, and $(N_t \times 256)$, (256×256), $(256 \times N_h)$ layers for medium and large respectively. This architecture is used to initially embed topological features, from vectorized PH. For the topo-net, used in TBN we use similar architecture design.

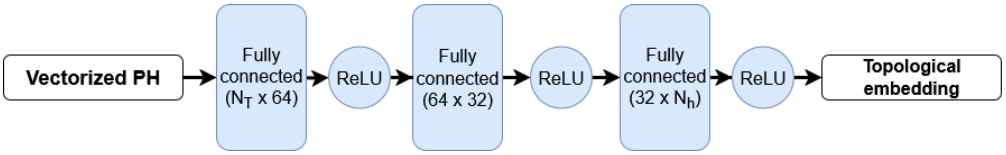

Figure 11: Architecture of the Topological Embedding Network (TEN) in small size.

Local Topological Embedding Network (LTEN) as depicted in Figure 12, used to further embed topological features to be added into each residual block of the backbone ResNet18 model. Size referred to as large uses the same number of layers as the medium size, with different number of neurons in each layer, that is $(N_h \times 128)$, (128×64), $(64 \times N_c)$ layers. Where N_h is the size of the hidden vector, and N_c is the number of output channels for the residual block for which we add the output of LTEN.

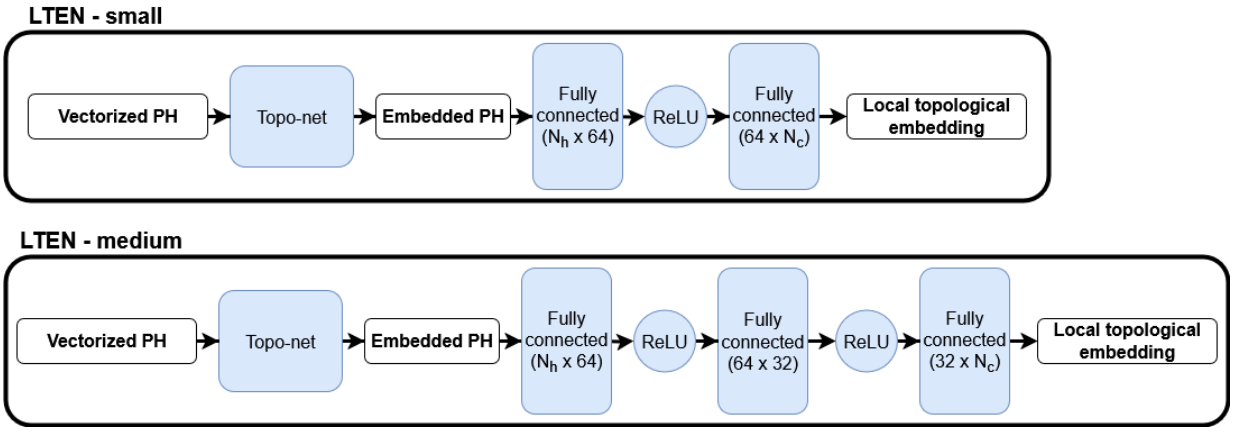

Figure 12: Architecture of the Local Topological Embedding Network (LTEN) for small and medium sizes.

Appendix B. PI Image TRN and TBN architectures

TRN architecture for persistent images as depicted in Figure 13, is used instead of previously mentioned architectures, since persistent images are 2D images, instead of plain vectors. Architecture design mostly mimics the backbone ResNet model, with different number of residual blocks. The sizes medium and large differ by the number of residual blocks in the topological processing.

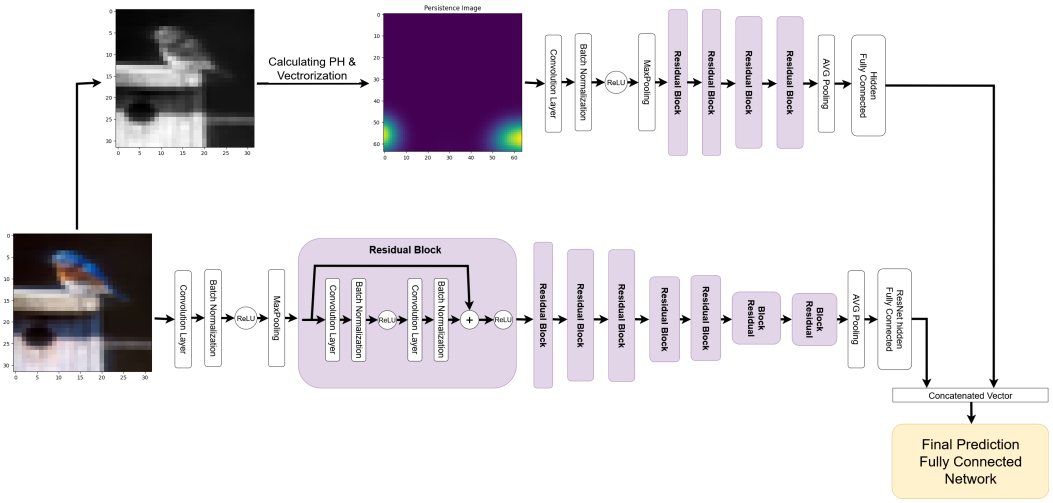

Figure 13: Architecture design for the Topological Residual Network for persistent images vectorization Topological features are processed separately from the Residual Network, only being concatenated and processed at the end.

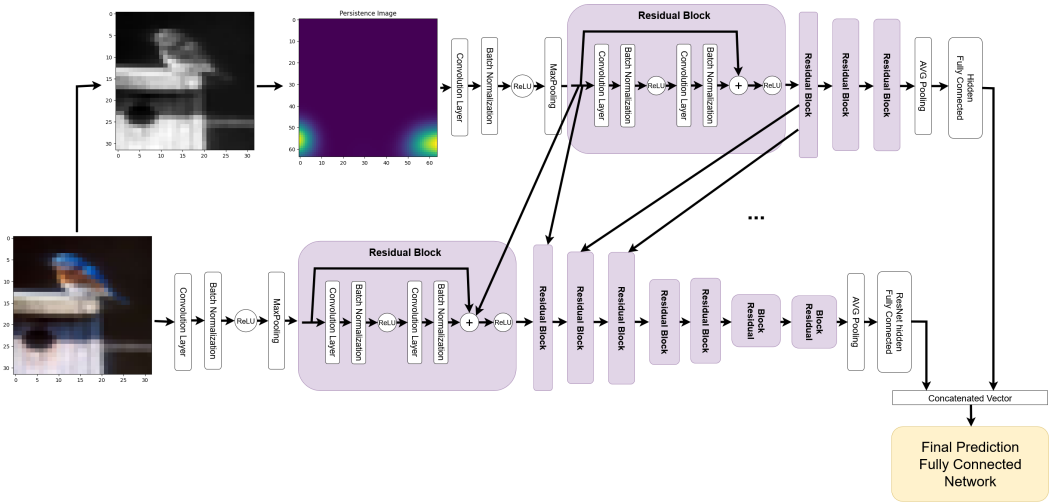

Figure 14: Architecture design for the Topological Block Network for persistent images vectorization. For clarity of the graph we do not show all connections between topological network and backbone residual blocks.

Similarly, TBN architecture for persistent images depicted in Figure 14 is used instead of previously mentioned TBN architecture. Note that the vector passed to backbone ResNet is taken directly from residual blocks responsible for embedding topological features. Therefore the part of the network that handles persistent images, mimics the architecture details of ResNet to appropriately add the information into residual blocks.

Appendix C. Aggregated augmentation settings results

This section presents the aggregated results for different augmentation settings tested, over the implementations, that achieved the highest validation set accuracy in the initial runs. For each vectorization method we chose at least one model for TRN and TBN architectures. We see no significant difference between different augmentation methods; however, a noticeable increase in performance when adding more perturbated samples into the dataset.

Augmentation	Min Acc (%)	Max Acc (%)	AVG Acc (%)	STD (%)
Topological	**83.76**	91.45	**87.84**	1.87
Non-Topological	82.78	91.01	87.73	1.84
All	82.78	**91.47**	87.50	1.98
30% additional samples	82.78	88.59	86.34	1.47
50% additional samples	**86.83**	**91.47**	**89.03**	1.18

Table 6: Aggregated results for the runs using different augmentation methods and the number of additional samples added for training.

Appendix D. Detailed CIFAR-10C results

We present the accuracy results across all available corruption types in the CIFAR-10C dataset. We note that all topological methods outperform the baseline ResNet on `elastic transform`, `fog`, `glass blur`, `impulse noise`, `JPEG compression`, and `pixelate` corruptions, while most of them still underperform the baseline in overall accuracy. Most of these corruption types do not interfere significantly in the homology groups created from the image data.

Corruption type	ResNet	BC-TBN-0	Land-TBN-1	PI-IMG-TRN-Concat	Silh-TBN-concat
Brightness	**71.1**	48.1	47.0	48.7	42.7
Contrast	**25.6**	18.5	22.32	24.8	18.7
Defocus-blur	**71.4**	58.1	63.4	64.5	51.9
Elastic-transform	53.2	64.7	67.7	**68.3**	64.1
Fog	44.0	48.4	**55.8**	54.4	45.9
Frost	52.6	47.5	53.0	**54.5**	49.2
Gaussian-noise	**42.9**	32.6	42.2	34.4	39.0
Glass-blur	47.3	56.0	**59.0**	58.0	57.8
Impulse-noise	27.6	30.9	**45.2**	40.0	41.6
JPEG-compression	59.6	69.6	**73.0**	71.23	68.6
Motion-blur	50.5	56.2	61.8	**61.9**	53.1
Pixelate	68.6	71.7	73.9	**75.2**	70.0
Shot-noise	**45.0**	34.6	42.0	36.5	38.8
Snow	**56.9**	39	40.6	43.5	36.6
Zoom-blur	60.4	61.9	**65.8**	65.6	58.2
Total	51.78	49.19	**54.17**	53.44	49.08

Table 7: Accuracy of the CIFAR-10C Dan Hendrycks [2019] benchmark runs for each corruption type available in the dataset.