PI-Net: A Deep Learning Approach to Extract Topological Persistence Images

Anirudh Som*, Hongjun Choi*, Karthikeyan Natesan Ramamurthy†, Matthew Buman†, Pavan Turaga*
*Geometric Media Lab, Arizona State University, Tempe, AZ, USA
†IBM Research, Yorktown Heights, NY, USA
{anirudh.som, hchoi71}@asu.edu, knatesa@us.ibm.com, {matthew.buman, pavan.turaga}@asu.edu

Abstract

Topological features such as persistence diagrams and their functional approximations like persistence images (PIs) have been showing substantial promise for machine learning and computer vision applications. Key bottlenecks to their large scale adoption are computational expenditure and difficulty in incorporating them in a differentiable architecture. We take an important step in this paper to mitigate these bottlenecks by proposing a novel one-step approach to generate PIs directly from the input data. We propose a simple convolutional neural network architecture called PI-Net that allows us to learn mappings between the input data and PIs. We design two separate architectures, one designed to take in multi-variate time series signals as input and another that accepts multi-channel images as input. We call these networks Signal PI-Net and Image PI-Net respectively. To the best of our knowledge, we are the first to propose the use of deep learning for computing topological features directly from data. We explore the use of the proposed method on two applications: human activity recognition using accelerometer sensor data and image classification. We demonstrate the ease of fusing PIs in supervised deep learning architectures and speed up of several orders of magnitude for extracting PIs from data. Our code is available at https://github.com/anirudhsom/PI-Net.

1. Introduction

Deep learning over the past decade has had tremendous impact in computer vision, natural language processing, machine learning, and healthcare. Among other approaches, convolutional neural networks (CNNs) in particular have received great attention and interest from the computer vision community. This is attributed to the fact that they are able to exploit the local temporal and spatial correlations that exist in 1-dimensional (1D) sequential time-series signals, 2-dimensional (2D) data like images, 3-dimensional (3D) data like videos, and 3D objects.

Figure 1. Illustration of the proposed PI-Net model to directly compute topological features called persistence images.

In this paper, we refer to these type of data as input data. CNNs also have far less learnable parameters than their fully-connected counterparts, making them less prone to over-fitting and have shown state-of-the-art results in applications like image classification, object detection, scene recognition, fine-grained categorization and action recognition [26, 20, 50, 51, 52]. Apart from being good at learning mappings between the input and corresponding class labels, deep learning frameworks are also efficient in discovering mappings between the input data and other output feature representations [45, 47, 28, 16, 13].

While methods for learning features from scratch and mapping data to desired outputs via neural networks have matured significantly, relatively less attention has been paid to invariance to nuisance low-level transforms like Gaussian noise, blur and affine transformations. Topological data analysis (TDA) methods are popularly used to characterize the shape of high-dimensional point cloud data using representations such as persistent diagrams (PDs) that are robust to certain types of variations in the data [14]. The shape of the data is quantified by properties such as connected components, cycles, high-dimensional holes, level-sets and monotonic regions of functions defined on the data [14]. Topological properties are those invariants that do not change under smooth deformations like stretching, bending and rotation, but without tearing or gluing surfaces. These attractive traits of TDA has renewed interested in this area for answering various fundamental questions, including those dealing with interpretation, generalization, model selection, stability, and convergence [19, 6, 34, 32, 18, 17].

A lot of work has gone into utilizing topological representations efficiently in large-scale machine learning [3, 5,
machine learning tools [3, 5, 35, 30, 33, 1, 40]. To alleviate the first problem, in this paper we propose a simple one-step differentiable architecture called PI-Net to compute the desired topological feature representation, specifically persistence images (PIs). To the best of our knowledge, we are the first to propose the use of deep learning for computing PIs directly from data.

Our motivation to use deep learning stems from its successful use to learn mappings between input data and different feature representations [45, 47, 28, 16, 13]. However, deep learning and TDA did cross paths before but not in the same context as what we propose in this paper. TDA methods have been used to study the topology [19, 6], algorithmic complexity [34], behavior [18] and selection [32] of deep learning models. Efforts have also been made to use topological feature representations either as inputs or fused with features learned using neural network models [12, 24, 7]. Later in Section 5, we too show experimental results on fusing generated PIs with deep learning frameworks for action recognition and image classification tasks.

3. Background

Persistence Diagrams: Consider a graph \( G = (\mathcal{V}, \mathcal{E}) \) constructed from data projected onto a high-dimensional point-cloud space. Here, \( \mathcal{V} \) is the set of \( |\mathcal{V}| \) nodes and \( \mathcal{E} \) denotes the neighborhood relations defined between the samples. Topological properties of the graph’s shape can be estimated by first constructing a simplicial complex \( S \) over \( G \). \( S \) is defined as \( S = (G, \Sigma) \), with \( \Sigma \) being a family of non-empty level sets of \( G \), with each element \( r \in \Sigma \) is a simplex [15]. This falls under the realm of persistent homology when we are interested in summarizing the \( k \)-dimensional holes present in the data. The simplices are constructed using the \( \epsilon \)-neighborhood rule [15]. It is also possible to quantify the topology induced by a function \( g \) defined on the vertices of a graph \( G \) by studying the topology of its sub-level or super-level sets. Since \( g : \mathcal{V} \rightarrow \mathbb{R} \), this is referred to as scalar field topology. In either case, PDs provide a simple way to summarize the birth vs death time information of the topological feature of interest. Birth-time (b) refers to the scale at which the feature was formed and death-time (d) refers to the scale at which it ceases to exist. The difference between \( d \) and \( b \) gives us the life-time or persistence and is denoted by \( l = |d - b| \).

Each PD is a multi-set of points \((b, d)\) in \( \mathbb{R}^2 \). Interested readers can refer to the following papers to learn more about the properties of the space of PDs [14, 15].

Persistence Images: A PI is a finite-dimensional vector representation of a PD [1] and can be computed through the following series of steps. First we map the PD to an integrable function \( \rho : \mathbb{R} \rightarrow \mathbb{R}^2 \) called a persistence surface. The persistence surface \( \rho \) is defined as a weighted sum of Gaussian functions that are centered at each point in the plane. The persistence images are finally computed using a differentiable neural network architecture called PI-Net, to extract topological representations. In this paper we focus on persistence images (PIs) as the desired topological feature. (2) We provide two simple CNN-based architectures called Signal PI-Net that takes in multi-variate 1D sequential data and Image PI-Net that takes in multi-channel 2D image data. (3) We also employ transfer learning strategies to train the proposed PI-Net model on a source dataset and use it on a target dataset. (4) Through our experiments on hand gestures, using accelerometer sensor data and image classification on standard image datasets, we show the effectiveness of the generated approximations for PIs and compare their performance to PIs generated using TDA approaches. We also explore the benefits of concatenating PIs with features learnt using deep learning methods like Alexnet [26], Network-in-Network [27] for image classification and test their robustness using a differentiable architecture.

Contributions: (1) We propose a novel differentiable neural network architecture called PI-Net, to extract topological representations. In this paper we focus on persistence images (PIs) as the desired topological feature. (2) We provide two simple CNN-based architectures called Signal PI-Net that takes in multi-variate 1D sequential data and Image PI-Net that takes in multi-channel 2D image data. (3) We also employ transfer learning strategies to train the proposed PI-Net model on a source dataset and use it on a target dataset. (4) Through our experiments on hand gestures, using accelerometer sensor data and image classification on standard image datasets, we show the effectiveness of the generated approximations for PIs and compare their performance to PIs generated using TDA approaches. We also explore the benefits of concatenating PIs with features learnt using deep learning methods like Alexnet [26], Network-in-Network [27] for image classification and test their robustness using a differentiable architecture.
the PD. Next, a discretization of a subdomain of the persistence surface is done which results in a grid. Finally, the PI is obtained by integrating the persistence surface over each grid box, giving us a matrix of pixel values. An interesting aspect when computing PIs is the broad range of weighting functions to chose from, to weight the Gaussian functions. Typically, points of high persistence or lifetime are perceived to be more important than points of low persistence. In such cases one may select the weighting function to be non-decreasing with respect to the persistence value of each point in the PD. Adams et al. also talk about the stability of persistence images with respect ot the 1-Wasserstein distance between PDs [1]. Figure 2 shows an example of a PD and its PI that is weighted by its life-time.

Convolutional Neural Networks: CNNs were inspired from the hierarchical organization of the human visual cortex [21] and consist of many intricately interconnected layers of neuron structures serving as the basic units to learn, extract both low-level and high-level features from images. CNNs are particularly more attractive and powerful compared to their connected counterparts because CNNs are able to exploit the spatial correlations present in natural images and each convolutional layer has far less trainable parameters than a fully-connected layer. Several sophisticated CNN architectures have been proposed in the last decade, for example AlexNet [26], Network-in-Network [27], VGG [38], GoogleNet [42], ResNet [22], etc. Some of these designs are known to surpass humans for object recognition tasks [37]. Apart from discovering features from scratch for classification tasks, CNNs are also popular for learning mappings between input and other feature representations [45, 47, 28, 16, 13]. This motivates us to design simple CNN models for the task of learning mappings between the data and their PI representations. We would like to direct interested readers to the following survey paper to know more about different CNN architectures [41].

Learning Strategies: Here we will briefly talk about the two learning strategies namely: supervised learning and transfer learning. We employ these strategies to train the proposed PI-Net model. Supervised Learning is concerned with learning complex mappings from \(X\) to \(Y\) when many pairs of \((x, y)\) are given as training data, with \(x \in X\) being the input data and \(y \in Y\) being the corresponding label or feature representation. In a classification setting \(Y\) corresponds to a fixed set of labels. In a regression setting, the output \(Y\) is either a real number or a set of real numbers. In this paper our problem falls under the regression category as we try to learn a mapping between the input data and its PI. Transfer Learning is a design methodology that involves using the learned weights of a pre-trained model that is trained on a source dataset \(D_s\) for the source task \(T_s\), to initialize the weights of another model that is fine-tuned using a target dataset \(D_t\) for the target task \(T_t\) [48]. When training a model, abstract feature representations are usually learnt in the initial and middle layers, whereas task-specific features are learnt in the final layers. With transfer learning we only re-train or fine-tune the last layers. This allows us to leverage the source dataset that the model was initially trained on. The is useful in cases where the target dataset is a lot less compared to the source dataset. However, transfer learning only works if the features learned for the source task are generalizable for the target task. In Section 4 we show how transfer learning is employed in our proposed framework.

4. PI-Net Framework

In this section we will first go through the steps to generate the ground-truth PIs and later discuss the proposed network architecture. The two PI-Net variants are illustrated in Figure 3. To generate PIs from multi-variate time-series signals we use Signal PI-Net and for multi-channel images we use Image PI-Net.

4.1. Generating Ground Truth Persistence Images

Data Pre-processing: For uni-variate or multi-variate time-series signals, we consider only fixed-frame signals, i.e. signals with fixed number of time-steps, and zero-center them. We standardize the train and test sets such that they have unit variance along each time-step. For images we enforce the pixel value range to be \([0, 1]\).

Computing Persistence Diagrams and Images: We use the Scikit-TDA python library [36] and use the Ripser package for computing PDs. We only focus on extracting PDs of scalar functions on data. When working with 1D sequential data, these offer a way to describe extremal points. For example local minimums give birth to a topological feature (more accurately a 0-order homology group) which then die at local maxima. From our initial investigations we were able to generate PIs which had features of high persistence. We can also extract PDs from an image considering the pixel values to be the function on a 2D plane. However, using this approach we were not able to observe PDs with high persistence features for the image. Instead, we vectorize each image along its rows to form a 1D signal. We then extract PDs for the 1D signal, just as we do for time-series data. For multi-channel color images, we vectorize each
Figure 3. Illustration of the proposed Signal PI-Net (top) and Image PI-Net (bottom) networks to generate PIs directly from the input data.

Figure 4. Illustration of transfer learning being used to train the Image PI-Net model. The model is first trained on the source dataset and the last layers are later fine-tuned using the target dataset.

color channel separately and then compute PDs. For example, we reshape a $32 \times 32 \times 3$ color image to get $1024 \times 3$. This small change allowed us to observe more richer PDs that have features with high persistence.

For computing PIs we used the Persim package in the Scikit-TDA toolbox. In all our experiments we set the grid size of the generated PIs to $50 \times 50$ and fit a Gaussian kernel function on each point in the PD. We weight each Gaussian kernel by the life-time of the point. For all time-series datasets we set the standard deviation of the Gaussian kernel to 0.25 and set the birth-time range to $[-10, 10]$. For image datasets we fix the standard-deviation to 0.05 and the birth-time range to $[0, 1]$. Once we compute PIs we normalize each PI by dividing by its maximum intensity value. This forces the intensity values in the PI to lie between $[0,1]$.

4.2. Network Architecture

Here, we describe the Signal PI-Net and Image PI-Net architectures. These models are shown in Figure 3 and are designed using Keras with tensorflow backend [9].

**Signal PI-Net:** The input to the network is a $t \times n$ dimensional time-series signal, where $t$ refers to the number of time-steps or frame size. For a uni-variate signal $n = 1$ and for a multi-variate signal $n > 1$. For our experiments in Section 5, $t = 100$ and $n = 3$. After the input layer, the encoder block consists of four 1D convolution layers. Except the final convolution layer, all other convolution layers are followed by batch normalization, ReLU activation and Max-pooling. The final convolution layer is followed by batch normalization, ReLU activation and Global-average-pooling. The number of convolution filters is set to 128, 256, 512 and 1024 respectively. However, the convolution kernel size is same for all layers and is set to 3 with stride set to 1. We use appropriate zero padding to keep the output shape of the convolution layer unchanged. For all Max-pool layers, we set the kernel size to 3 and stride to 2. After the encoder block, we pass the global-average-pooled output into a final output dense layer of size $50 \times 50 \times n$. The output of the dense layer is subjected to ReLU activation and reshaped to size $50 \times 50 \times n$. As mentioned earlier, we set the height and width of all generated PIs to 50.

**Image PI-Net:** The input to this network is a $h \times w \times c$ dimensional image, where $h, w, c$ are the image’s height, width and number of channels. The structure of the encoder block is the same as that of the Signal PI-Net model. The only difference is that we now use the 2D version of the same layers described earlier. We pass the output of the encoder block into a latent variable layer which consists of a dense layer of size 2500. The output of the latent variable layer is reshaped to $50 \times 50 \times n$. As mentioned earlier, we set the height and width of all generated PIs to 50. To employ transfer learning, we first train the Image PI-
Second, we show improvements for image classification on both the GeneActiv (left) and USC-HAD (right) datasets.

Loss function: The Mean-Squared-Error loss function is used to quantify the deviation of the generated PIs from the ground-truth PIs. The different train and test loss trends for both Signal and Image PI-Net variants is shown in Figure 4.

### 5. Experiments

This section can be broadly divided into four parts. First we show human activity recognition on two accelerometer sensor datasets: GeneActiv [46] and USC-HAD [49]. Second, we show improvements for image classification task after fusing PIs obtained traditionally and using the proposed Image PI-Net framework with popular neural network architectures like Alexnet [26] and Network-in-Network [27]. For image classification we use the following datasets: CIFAR10 [25] and SVHN [29]. Third, we show how the generated PIs can be used to help improve robustness of deep learning models to different noises like blur, translation and Gaussian noise. Finally, we show improvements in computation time for the task of extracting PIs using the proposed method.

#### 5.1. Action Recognition using Accelerometer Data

We conduct this experiment on the following accelerometer datasets: GeneActiv [46] and USC-HAD [49]. The GeneActiv dataset consists of 29 different human-activity classes from 152 subjects. The data was collected at a sampling rate of 100Hz using a GeneActiv sensor, a light-weight, waterproof, wrist-worn tri-axial accelerometer. Please refer to the following paper to know more about the data collection protocol [46]. We extract non-overlapping frames of 10 seconds each, giving us about 31,275 frames. Each frame has a 1000 time-steps. We roughly use 75% of the frames for the training-set and the rest as test-set. To avoid inducing any bias, we make sure to place all frames from the same subject into either one of the sets. The USC-HAD dataset consists of 12 different human-activity classes from 14 subjects. It was collected using the tri-axial MotionNode accelerometer sensor at a sampling rate of 100Hz, with the sensor being placed at the front right hip [49]. Here also we extract 10 second non-overlapping frames resulting in about 2,499 frames. We use frames from the first 8 subjects for the training set and the remaining frames as the test set. Figure 5 show the list of all activity classes and their distribution for both datasets.

#### Training Signal PI-Net:
We train one Signal PI-Net model described in Section 4.2 using just the training set of the GeneActiv dataset. We set the batch-size to 128 and train the model for a 1000 epochs. The learning rate for the first 300 epochs, second 300 epochs and final 400 epochs was set to $10^{-3}$, $10^{-4}$ and $10^{-5}$ respectively. The Adam optimizer was used for training the model. We use the Mean-Squared-Error loss function to quantify the overall deviation of the generated PIs from the ground-truth PIs. The training and test loss trends are shown in Figure 6.

For characterizing the time-series signals, we consider three different feature representations: (1) A 19-dimensional feature vector consisting of different statistics calculated over each 10-second frame [46]; (2) Features learnt from scratch using 1D CNNs; (3) Persistence Images generated using the traditional filtration technique and the proposed Signal PI-Net model. The 19-dimensional feature vector includes mean, variance, root-mean-square value of the raw accelerations on each of X, Y and Z axes, pearson correlation coefficients between X-Y, Y-Z and X-Z time series, difference between maximum and minimum accelerations on each axis denoted by $dx, dy, dz$, and $\sqrt{dx^2 + dy^2}$, $\sqrt{dy^2 + dz^2}$, $\sqrt{dx^2 + dz^2}$, $\sqrt{dx^2 + dy^2 + dz^2}$. From here on out we will refer to this 19-dimensional statistics feature as SF. We use the trained Signal PI-Net model to extract PIs for the test set of the GeneActiv dataset. We also use the same model to compute PIs for both the training and test sets of the USC-HAD dataset. We wanted to see if we could exploit the knowledge learnt by the proposed Signal PI-Net model on a source dataset.

| Method                  | GeneActiv       | USC-HAD         |
|------------------------|-----------------|-----------------|
| MLP - PI               | 46.27 ± 0.28    | 44.71 ± 1.26    |
| MLP - Signal PI-Net    | 49.76 ± 0.90    | 48.21 ± 1.42    |
| MLP - SF [46]          | 33.48 ± 0.50    | 31.86 ± 2.47    |
| MLP - SF + PI          | 47.63 ± 0.43    | 45.79 ± 0.33    |
| MLP - SF + Signal PI-Net | 49.68 ± 0.22 | 48.68 ± 0.63 |
| 1D CNN                 | 56.34 ± 0.89    | 53.33 ± 1.35    |
| 1D CNN + PI            | 58.68 ± 0.49    | 55.67 ± 1.03    |
| 1D CNN + Signal PI-Net | 59.42 ± 0.35    | 58.56 ± 0.81    |

Table 1. Weighted F1 score classification results for the GeneActiv and USC-HAD datasets. The results shown are the mean ± standard-deviation values calculated over 5 runs.

![Figure 5. Distribution of the different human-activity classes for the GeneActiv (left) and USC-HAD (right) datasets.](image)
Figure 6. Train - Test loss trends for different PI-Net models. Here, we only show the trends from epochs 25 to 525. The train loss for the Signal PI-Net model starts at about 0.045 and for most Image PI-Net models at around 0.25. Between the epochs 1 and 1000 we see a reduction in the overall loss by about one order of magnitude in all cases.

Figure 7. Illustration of sample test images in CIFAR10 (1st row) and their corresponding ground-truth PIs (2nd row), PIs generated using the Image PI-Net model (3rd and 4th rows).

Figure 8. Illustration of sample test images in SVHN (1st row) and their corresponding ground-truth PIs (2nd row), PIs generated using the Image PI-Net model (3rd, 4th rows).

(GeneActiv) and use it on a target dataset (USC-HAD). As seen from Figure 5, there is a huge shift in both the data distribution and end-target classes. This pushes it to the realm of a cross-domain and cross-task learning problem. Cross-domain since for each dataset the accelerometer sensor was placed on different parts of the human body; and cross-task since the class-distribution and end classification task is very different for both datasets.

The weighted F1 score classification results is shown in Table 1. We use a multi-layer-perceptron (MLP) classifier for the SF, PI features and a 1D CNN classifier for the time-series signals. The MLP classifier contains 8 dense layers with ReLU activation and having 1024, 1024, 512, 512, 256, 256, 128, 128 units respectively. To avoid overfitting, each dense layer is followed by a dropout layer with a dropout rate of 0.2 and a batch-normalization layer. The output layer is another dense layer with Softmax activation and with number of units equal to the number of classes. The 1D CNN classifier consists of 10 CNN layers with number of filters set to 64, kernel size to 3, stride to 1 and the output is zero-padded. Each CNN layer is followed by batch-normalization, ReLU activation and max-pooling layers. For max-pool layers we set the filter size to 3, the stride was set to 1 for every odd layer and 2 for every even layer. For the final CNN layer we use a global-average-pooling layer instead of a max-pool layer. Here too, the output layer consists of a dense layer with softmax activation and number of units equal to number of target classes.

Table 1 shows results for both individual features and different fusion cases. In the table, PI refers to PIs obtained using conventional TDA methods and Signal PI-Net means PIs computed using the proposed Signal PI-Net model. We fuse SF and PI features at the input layer before passing into the MLP classifier. For 1D CNNs we fuse the PI features after the global-average-pooling layer. We see improvements in classification results using the proposed Signal PI-Net model for both datasets. We would like to remind our readers that the results for USC-HAD was obtained using the Signal PI-Net model trained on just the GeneActiv dataset. This opens doors to further explore the proposed framework on cross-domain, cross-task learning problems. For the 1D CNN case, apart from improving the overall classification accuracy we also notice the standard deviation being reduced after combining PIs. We further provide the confusion matrices for a few of the methods listed in table 1 in the Appendix at the end of the paper.

5.2. Image Classification

We use the following three image datasets for purposes of training different Image PI-Net models: CIFAR10, CIFAR100 [25] and SVHN [29]. However, we show image
Network-in-Network (NIN) models.

classification results for only CIFAR10 and SVHN. Both CIFAR10 and CIFAR100 contain 60,000 color images, which are split into 50,000 training images and 10,000 test images. The SVHN dataset contains 73,257 training images and 26,032 test images. Images have the same shape in all three datasets. The height, width, number of channels for each image is equal to 32, 32 and 3 respectively. Sample test images in each class for the CIFAR10 and SVHN dataset are shown in Figures 7 and 8 respectively.

Training Image PI-Net: We develop two kinds of Image PI-Net models based on the datasets we chose as source and target datasets to train the model: (1) In the first kind we set the source and target datasets to be same, i.e. we train the Image PI-Net model using the CIFAR10 or SVHN dataset. (2) For the second type, we use the CIFAR100 dataset as our source dataset and the target datasets are either CIFAR10 or SVHN. Simply put, we employ transfer learning by first training the Image PI-Net model using CIFAR100 and later use the target dataset to fine-tune the last layers as illustrated in Figure 4. For the second case, we further explore two variations: (2a) Fine-tune the last layers using all samples from the training set of the target dataset; (2b) fine-tune using just a subset i.e. 500 images per class in the training set of the target dataset. We will refer to these variants as Image PI-Net FA (Fine-tune All) and Image PI-Net FS (Fine-tune Subset) respectively. For all cases we normalize the images by dividing all pixels by 255. This scales all pixels to lie in the range [0, 1]. For the model described in Section 4.2 we set the batch-size to 128 and train the model for a 1000 epochs. Just like the Signal-To-PI model we set the learning rate for the first 300 epochs, next 300 epochs and final 400 epochs to $10^{-3}$, $10^{-4}$ and $10^{-5}$ respectively. Here too we use Adam optimizer and the Mean-Squared-Error loss function. The training and test loss trends for the different Image PI-Net models are shown in Figure 6.

Ground-truth PIs and PIs generated using the above Image PI-Net cases for both the CIFAR10 and SVHN are shown in Figures 7 and 8 respectively. For image classification we use Alexnet [26] and Network-in-Network (NIN) [27] as our base models. Topological features like PIs alone are not as powerful as features learnt by most deep learn-

### Table 2. The architecture descriptions for Alexnet (left) and Network-in-Network (right) models.

| Name | Description | Name | Description |
|------|-------------|------|-------------|
| input | 32 × 32 RGB image | input | 32 × 32 RGB image |
| conv1a | 32 filters, 3 × 3, pad=same', ReLU | conv1a | 64 filters, 3 × 3, ReLU |
| conv1b | 64 filters, 3 × 3, ReLU | conv1b | 64 filters, 3 × 3, ReLU |
| pool1 | Maxpool 2 × 2 | pool1 | Maxpool 2 × 2 |
| drop1 | Dropout 0.2 | drop1 | Dropout 0.2 |
| conv2a | 128 filters, 3 × 3, pad=same', ReLU | conv2a | 128 filters, 3 × 3, pad=same', ReLU |
| conv2b | 128 filters, 3 × 3, ReLU | conv2b | 128 filters, 3 × 3, ReLU |
| pool2 | Maxpool 3 × 3, stride=(2,2), pad=same | pool2 | Maxpool 3 × 3, stride=(2,2), pad=same |
| drop2 | Dropout 0.2 | drop2 | Dropout 0.2 |
| dense1 | Fully connected 1024 units, ReLU | dense1 | Fully connected 1024 units, ReLU |
| drop3 | Dropout 0.2 | drop3 | Dropout 0.2 |
| dense2 | Fully connected 256 units, ReLU | dense2 | Fully connected 256 units, ReLU |
| output | Softmax | output | Softmax |

Table 3. Image classification results for CIFAR10 and SVHN datasets. Mean±Standard-deviation values are shown over 5 runs. p-values are calculated with respect to the base model.

| Method | CIFAR10 | SVHN |
|--------|---------|------|
| Alexnet | 80.49±0.30 | 93.08±0.17 |
| Alexnet + PI | 80.52±0.38 | 93.72±0.10 |
| Alexnet + Image PI-Net | 81.25±0.49 | 93.83±0.11 |
| Alexnet + Image PI-Net FA | 81.23±0.42 | 93.92±0.13 |
| Alexnet + Image PI-Net FS | 81.80±0.24 | 93.94±0.13 |
| NIN | 80.29±0.30 | 95.83±0.07 |
| NIN + PI | 85.29±0.30 | 95.75±0.08 |
| NIN + Image PI-Net | 86.61±0.19 | 96.04±0.04 |
| NIN + Image PI-Net FA | 86.61±0.39 | 96.07±0.05 |
| NIN + Image PI-Net FS | 86.61±0.40 | 96.06±0.04 |

Figure 9. Illustration of the modified base model where we concatenate PI feature with features learnt using the base classification network.

5.3. Robustness to Noise

In this section we create noisy variations of the test set in CIFAR10 and SVHN. In particular we generate noisy images...
We used 4 NVIDIA GeForce GTX Titan Xp graphic cards, each with 12GB memory to train and evaluate all deep learning models. All our tasks were carried out on a standard Intel i7 CPU using Python with a working memory of 32GB. We used the Scikit-TDA software to compute PDs and PI(s) [36]. Table 5 shows the average time taken by conventional TDA methods using one CPU and the proposed Image PI-Net framework on just one GPU, to extract PI for one image. The average is computed over all images present in the training set of the dataset. Using the Image PI-Net model, we see an effective speedup of up to two orders of magnitude in the computation time. We also check the time taken to compute PDs when the entire training set is passed into the Image PI-Net model as a single batch. For the entire training set it takes around 9.77±0.08 seconds for CIFAR10 and 12.93±0.05 seconds for SVHN. This is a fraction of the time compared to the time it takes using conventional TDA tools. So far it had been impossible to compute PDs at real-time using conventional TDA approaches. However, the proposed framework allows us to easily compute PDs in real-time thereby opening doors to new real-time applications for TDA.

6. Conclusion and Future Work
In this paper we took the first step in using deep learning to extract topological feature representations. We developed a differentiable and effective architecture called PI-Net to extract PDs directly from data. PI-Net has a significantly lower computational complexity compared to using conventional topological tools. We show good results on different time-series and image datasets, and also test the robustness of different base classification networks together with PI(s) generated using PI-Net for different kinds of noise added to the data.
icated deep learning architectures that can allow us to learn mappings between higher dimensional data and their corresponding topological feature representations. We would also like to see how deep learning can be further used to generate other kinds of topological representations. Also, conventional TDA tools are invariant to small perturbations on the input data space. Now that we are able to generate approximations of topological representations, it would be interesting to use the proposed framework in a setting that is resistant to adversarial attacks which is major issue faced by current deep neural networks.

7. Acknowledgements

This work was supported in part by NSF CAREER grant 1452163 and ARO grant number W911NF-17-1-0293.

References

[1] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. Journal of Machine Learning Research, 18(8):1–35, 2017. 2

[2] H. Adams, A. Tausz, and M. Vejdemo-Johansson. Javaplex: A research software package for persistent (co) homology. In International Congress on Mathematical Software, pages 129–136. Springer, 2014. 2

[3] R. Anirudh, V. Venkataaraman, K. Natesan Ramamurthy, and P. Turaga. A riemannian framework for statistical analysis of topological persistence diagrams. In The IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 68–76, 2016. 2

[4] U. Bauer, M. Kerber, and J. Reininghaus. Distributed computation of persistent homology. In Proceedings of the Workshop on Algorithm, Engineering and experiments, pages 31–38. SIAM, 2014. 2

[5] P. Bubenik. Statistical topological data analysis using persistence landscapes. The Journal of Machine Learning Research, 16(1):77–102, 2015. 2

[6] P. Bubenik and J. Holcomb. Statistical inferences from the topology of complex networks. Technical report, Cleveland State University, Cleveland, United States, 2016. 1, 2

[7] Z. Cang and G.-W. Wei. Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. PLoS Computational Biology, 13(7), 2017. 2

[8] H. Chintakunta, T. Gentimis, R. Gonzalez-Diaz, M.-J. Jimenez, and H. Krim. An entropy-based persistence barcode. Pattern Recognition, 48(2):391–401, 2015. 2

[9] F. Chollet et al. Keras. https://keras.io, 2015. 4

[10] M. K. Chung, P. Bubenik, and P. T. Kim. Persistence diagrams of cortical surface data. In International Conference on Information Processing in Medical Imaging, pages 386–397. Springer, 2009. 2

[11] Y. Dabaghian, F. Mémoi, L. Frank, and G. Carlsson. A topological paradigm for hippocampal spatial map formation using persistent homology. PLoS Computational Biology, 8(8):1–14, 2012. 2

[12] T. K. Dey, S. Mandal, and W. Varcho. Improved Image Classification using Topological Persistence. In Vision, Modeling & Visualization. The Eurographics Association, 2017. 2, 7

[13] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In European Conference on Computer Vision, pages 184–199. Springer, 2014. 1, 2, 3

[14] H. Edelsbrunner and J. Harer. Computational topology: an introduction. American Mathematical Society, 2010. 1, 2

[15] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. Discrete & Computational Geometry, 28(4):511–533, 2002. 2

[16] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In Advances in Neural Information Processing Systems, pages 2366–2374, 2014. 1, 2, 3

[17] M. Ferri. Why topology for machine learning and knowledge extraction? Machine Learning and Knowledge Extraction, 1(1):115–120, 2018. 1

[18] M. Gabella, N. Afambo, S. Ebli, and G. Spreemann. Topology of learning in artificial neural networks. arXiv preprint arXiv:1902.08160, 2019. 1, 2

[19] R. B. Gabrielsson and G. Carlsson. Exposition and interpretation of the topology of neural networks. arXiv preprint arXiv:1810.03234, 2018. 1, 2

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587, 2014. 1

[21] K. Grill-Spector and R. Malach. The human visual cortex. Annu. Rev. Neurosci., 27:649–677, 2004. 3

[22] 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. 3

[23] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In IEEE Conference on Computer Vision and Pattern Recognition, 2010. 2

[24] C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl. Deep learning with topological signatures. In Advances in Neural Information Processing Systems, pages 1634–1644, 2017. 2

[25] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5, 6

[26] 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, 3, 5, 7

[27] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013. 2, 3, 5, 7

[28] 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. 1, 2, 3
Appendix

Confusion Matrix for Time-series Human Activity Recognition

Here we show the confusion matrices for a few of the methods listed in Table 1. Specifically we show the confusion matrix for the Multi Layer Perceptron (MLP) method on the 19-dimensional statistical feature (SF), persistence image (PI) obtained using conventional topological data analysis (TDA) tools and PI computed using the proposed PI-Net model. We show these for both the GeneActiv [46] and USC-HAD [49] datasets in Figures 10 and 12 respectively. We also show the confusion matrices for the 1-dimensional convolutional neural network (1D CNN) both alone and in fusion with the two PI variants in Figures 11 and 13 respectively. For the MLP classifier we observe PI features being more informative than the SF method. We also observe that fusing PIs with more powerful classifiers like 1D CNNs helps improve the overall classification performance. We used the same Signal PI-Net model trained on the GeneActiv dataset to extract PIs for the USC-HAD dataset, i.e. we do not fine-tune the model again using the USC-HAD dataset.
Figure 10. Confusion matrices for MLP - SF [46] (left), MLP - PI (middle) and MLP - Signal PI-Net (right) methods on the GeneActiv dataset [46] listed in Table 1.

Figure 11. Confusion matrices for 1D CNN (left), 1D CNN + PI (middle) and 1D CNN + Signal PI-Net (right) methods on the GeneActiv dataset [46] listed in Table 1.

Figure 12. Confusion matrices for MLP - SF (left), MLP - PI (middle) and MLP - Signal PI-Net (right) methods on the USC-HAD dataset [49] listed in Table 1.
Figure 13. Confusion matrices for 1D CNN (left), 1D CNN + PI (middle) and 1D CNN + Signal PI-Net (right) methods for on USC-HAD dataset [49] listed in Table 1.