TDAsweep: A Novel Dimensionality Reduction Method for Image Classification Tasks

Yu-Shih Chen
Department of Computer Science
University of California, Davis
Davis, CA 95616

Melissa Goh
Oracle Corporation
Seattle, WA 98101

Norm Matloff (corresponding author)
Department of Computer Science
University of California, Davis
Davis, CA 95616

December 6, 2020

1 Summary

One of the most celebrated achievements of modern machine learning technology is automatic classification of images. However, success is typically achieved only with major computational costs. Here we introduce **TDAsweep**, a machine learning tool aimed at improving the efficiency of automatic classification of images.

2 Motivation

Machine learning technology has become ubiquitous, in business, engineering, the sciences, medicine and so on. One of the most visible types of applications is image classification. Much research is being conducted for instance in medical applications, such as detection of cancer from histology slide data.

In principle, image classification should not be different from any other classification application. Any of the usual methods, say logistic regression, support vector machines or neural networks might be used, taking as features the pixel values. But in practice, the situation quickly spirals out of control as the image size grows. Even with the famous MNIST image test bed [7], consisting of very small 28 x 28 images, such an approach implies that the number of features is $28^2 = 784$. Hence an urgent need for *dimension reduction*, i.e. reducing the number of features.

A popular form of dimension reduction in imaging settings is use of *convolutional* methods, which adapt classical image tiling techniques [3]. With proper tuning, these can work well, but at a cost of extensive computation time and large memory footprint. Thus an alternative approach to dimensional reduction is desirable. Here we introduce a new method, TDAsweep, along with software implementing it.

3 Topological Data Analysis

TDAsweep is a variant of *topological data analysis* (TDA) [2], a feature extraction technique. In the case of image analysis, TDA forms a *filtration*, a family of geometric overlays of the image. typically involves pixel intensity thresholding, as follows:

Consider *r x s* images in which pixel intensity varies from 0 to 255 (for simplicity, assumed here to be grayscale). Set a threshold *t*, say 100. At each pixel whose intensity is below *t*, set the intensity to 0. The remaining nonzero pixels then will clump into one or more connected sets, in which two pixels are in the same set if they are adjacent horizontally, vertically or diagonally. As we vary *t*, connected sets can grow, shrink, coalesce and subdivide, with the set counts growing and shrinking correspondingly. These counts, *Betti numbers*, form our new, dimension-reduced feature set [4].

4 TDAsweep

Our **TDAsweep** method is inspired both by TDA and by *Run Length Run Number* (RLRN) methods [5] [6]. In RLRN, the counts are of the numbers of consecutive pixels above a threshold, within a given direction.

TDAsweep should be viewed as a blend between standard TDA and RLRN. For a given threshold t and a given one of the six directions, we count the number of connected components. These counts form the new features.

TDAsweep casts thresholding on the original image (each pixel value above the threshold will be denoted as 1, 0 otherwise). Then, TDAsweep counts contiguous components in horizontal, vertical, and the two diagonal directions of a pixel matrix. The counts of the components in each direction will serve as the new set of features describing the original image.

4.1 Toy Example

An example should help illustrate the process more clearly: After thresholding some toy image, say we have the following matrix:

```
10011101
10111100
10101101
```

Then, we would count the number of components in each row, column, and diagonal. An example of counting the components for each row would be:

```
There are 3 components in vector 10011101
There are 2 components in vector 10111100
There are 4 components in vector 10101101
```

Here, [3,2,4] will be included as the new set of features. We repeat this process for columns and the two diagonal directions (NW to SE and NE to SW), for each *t*.

5 Goals of the Method and Software

TDAsweep is aimed at reducing computation time while retaining classification accuracy. By first performing dimension reduction using **TDAsweep**, it is hoped to attain a major reduction in the subsequent model fit time.

Tuning Parameters

Compared to the "C" part of CNN, **TDAsweep** has rather few tuning parameters/hyperparameters:

- the threshold values (number and levels)
- intervalWidth: a coalescing factor

The latter parameter controls further dimension reduction beyond the initial sweep operation. Say we have an *r x s* image, with this parameter set to *w*. Partition the rows into *r/w* groups of consecutive rows. Then within each group, replace the *w* component counts by just their sum. Do the same for columns and diagonals.

5.1 Parallelization Feature

TDAsweep provides the user with the option to parallelize the process. To enable the feature, simply specify the desired number of cores, via the arument *cls*. If parallelization is not needed, set *cls=NULL*. Comparisons of run-time difference between parallel and non-parallel TDAsweep is shown in the showcases below.

6 Example: MNIST Data

MNIST is an image dataset consisting of 28x28 grayscale handwritten digit images; It is one of the most widely known image datasets for classification.

As one of the case studies, we compared the results of using only the Support Vector Machine (SVM) without any form of dimensionality reduction and the results of using *TDAsweep* before SVM for classification tasks on *MNIST* (refer to Table 1). We used the standard **e1071** package in R with the default hyperparameters.

Table 1: **SVM + TDAsweep on MNIST.** A comparison between the performance of the SVM model with and without TDAsweep as dimensionality reduction and feature extraction.

	MNIST (raw)	**MNIST (sweep)**	**MNIST (sweep)**
No. of features	784	84	168
intervalWidth	NA	2	2
thresholds	NA	(100)	(100,175)
TDAsweep total time	NA	42 min	78 min
TDAsweep (core=4)	NA	12.4 min	21.7 min
SVM train time	123 min	14 min	24 min
Accuracy (Validation Set)	97.9%	97%	97.86%

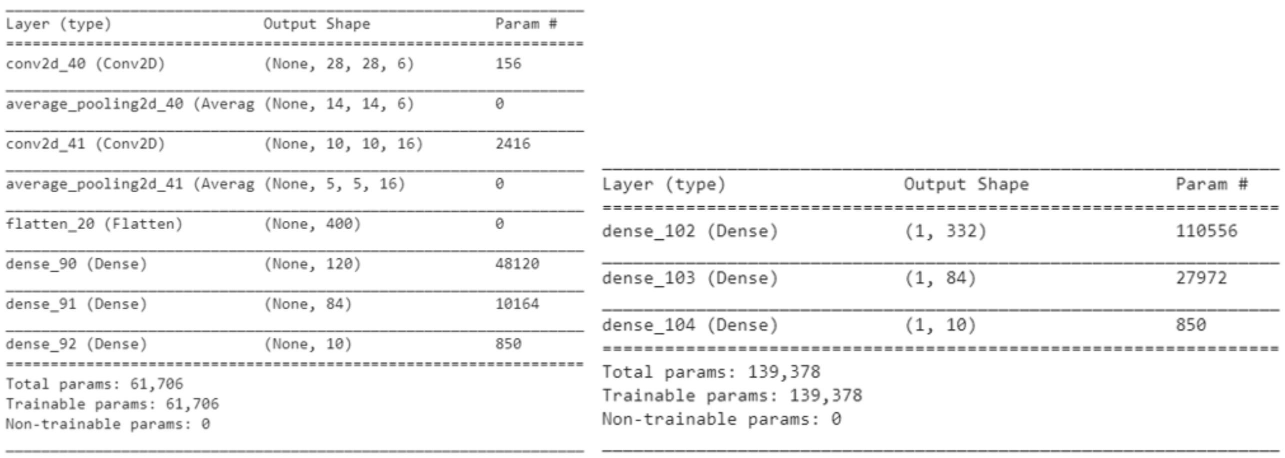
As we can see from Table 1, with one threshold and **intervalWidth** equal to 2, we were able to achieve 89.3% feature reduction (784 features to 84 features) in exchange for less than 1% accuracy loss. With some tuning, specifically with 2 thresholds, we achieved a 78.6% feature reduction in exchange for a mere 0.04% accuracy loss. **TDAsweep** drastically decreased the SVM training time, from 123 minutes to 14 and 24 minutes for 1 and 2 thresholds, with a corresponding reduction in overall time from 123 minutes to 56 minutes and 102 minutes, respectively.

7 Further Investigation: TDAsweep Versus Le-Net 5

The convolution operation in CNN is essentially another feature extraction and reduction method for the dense layers, hence it would be interesting to compare our method with the 'C' in CNN. We have found our method to be very effective on medical image sets, specifically on Histology images,and we show a case study where TDAsweep + dense neural network out-performs Le-Net 5, a popular CNN architecture known for its simplicity and high-performance.

7.1 Experiment Setup

The following shows the architecture for Le-Net 5 and the dense layer for TDAsweep (Figure 1).



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_40 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d_40 (Averag (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_41 (Averag (None, 5, 5, 16)          0
_________________________________________________________________
flatten_20 (Flatten)         (None, 400)               0
_________________________________________________________________
dense_90 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_91 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_92 (Dense)             (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_102 (Dense)            (1, 332)                  110556
_________________________________________________________________
dense_103 (Dense)            (1, 84)                   27972
_________________________________________________________________
dense_104 (Dense)            (1, 10)                   850
=================================================================
Total params: 139,378
Trainable params: 139,378
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: **Structure of Le-Net 5** Figure generated with the keras package in Python.

For training both networks, we used the Adam optimizer with a learning rate of 5e-4, a sparse categorical crossentropy for the loss function, and 50 epochs (with an early stop feature) for both Le-Net 5 and the dense layer for TDAsweep. For TDAsweep, we used thresh=[25,100] and intervalWidth=1, resulting in 332 new features after reduction.

7.2 Training Results & Discussion

We trained both Le-Net 5 and TDAsweep + dense layer 20 times and plotted a box-plot of their test accuracy for comparison.

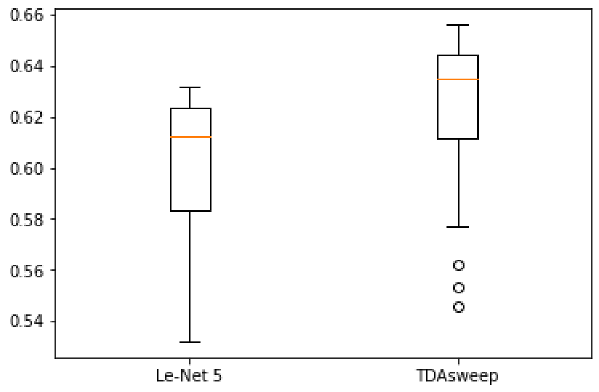


Figure 2: **Boxplot comparison between TDAsweep and Le-Net 5** Each box plot shows the test accuracy of the Histology image dataset trained with Le-Net5 and TDAsweep, respectively.

We show some basic statistics such as mean, standard deviation, minimum, and maximum of the test accuracy over the 20 trains.

Table 2: **Basic statistics over test accuracy of 20 training attempts.**

	mean	**std**	**min**	**max**
Le-Net 5	0.6000	0.0277	0.5320	0.6320
TDAsweep + NN	0.6214	0.0347	0.5460	0.6560

As we could clearly see, after 20 training attempts, TDAsweep + NN still out-performs the Le-Net 5 CNN by around 2%.

8 Experiments & Characteristic Analysis

The above timings were for the **e1071** R package implementation of SVM. This is the "standard" implementation, but a lesser-known package, **liquidSVM**, can be much faster. The experiments described below were conducted with this version of SVM. We have used the default parameters for the **liquidSVM** package as well.

Other than *MNIST*, we analyzed the multifaceted advantages that *TDAsweep* could bring with other widely known image datasets such as *histology-MNIST* [**?**] and *CIFAR-10* [9].

9 TDAsweep Versus Image Augmentation

Image augmentation is a technique used to generate new image sample data from the existing ones, through random geometric transformations such as horizontal and vertical flips, rotations, scale transformations and so on. They are of particular interest in applications in which the images have no inherent up/down, left/right orientation, such as those from histology slides. The reasoning is that a new image to be classified may be similar to a rotated version, say, of one of the images in the training set, even though not similar to any of those images themselves.

Data augmentation in image processing became well-known after the development of AlexNet [1]. Numerous augmented files were added, multiplying the number of images by a factor of 2048. The resulting success inspired widespread usage of the technique.

However, one advantage of **TDAsweep** is that to a large extent it obviates the need for data augmentation. For example, consider a situation of vertical flipping. Since the column component counts are the same in a flipped version of an image as in the original, there is no need to add the flipped version to the dataset. The same is true for horizontal flips and row component counts and so on.

Since data augmentation can greatly increase computation time for SVM or whatever other machine learning algorithm is used for classification, this can be a major win for **TDAsweep**. The more augmented files are added, the longer the algorithm run time, thus the greater the potential advantage of **TDAsweep**.

(However, in the Tensorflow and Pytorch approach, the software does not actually add new images. Instead, at the batch level, the data are replaced by randomly-transformed versions. This is done at every iteration. Computation time is not appreciably increased under this approach.)

9.1 Example: Histology Images

The following shows an example of TDAsweep + liquidSVM on the raw histology-MNIST dataset compared to just the liquidSVM on an augmented histology-MNIST dataset (refer to Table 3). For this experiment, the original histology-MNIST dataset consists of 5000 grayscale histology images, each 64 by 64 in size. We obtained the augmented version of histology-MNIST dataset by running data augmentation through flipping, rotation, and shifting, reaching a total sample size of 120,000 histology images, a data expansion factor of 24.

Table 3: **Example of TDAsweep versus image augmentation.** A comparison between the performance of the TDAsweep and image augmentation technique on the Histology Images, each 64x64 in size.

	MNIST (raw)	**MNIST (sweep)**	**MNIST (sweep)**
Sample size	5,000	120,000	5,000
No. of features	4096	4096	382
intervalWidth	NA	NA	1
thresholds	NA	NA	(25,100)
TDAsweep total time	NA	NA	169 min
TDAsweep (core=4)	NA	12.4 min	103.76 min
SVM train time	1 min	310.26 min	0.42 min
Accuracy (Validation Set)	57%	57.24%	70.01%

Let us examine the results in Table 3. Since 5000 is a relatively small sample size, naturally, one would look into data augmentation to increase the data's variation. However, we notice that, even after data augmentation increased the sample size to 120,000, liquidSVM was only able to raise the accuracy by 0.24

As we can see from the results, running **TDAsweep** on the original dataset with 5000 samples improved the accuracy by a significant amount (roughly 13% increase), showing how well **TDAsweep** extracted insightful features from the histology images used in this experiment.

Moreover, **TDAsweep** + liquidSVM on the original dataset also had a large run-time advantage over liquidSVM on the augmented dataset, 169.42 minutes versus 310.26 minutes. This example demonstrates the strong potential of **TDAsweep** as a better alternative for smaller datasets requiring data augmentation, both in run-time and accuracy. Note that if our expansion factor had larger than 24, more like the 2048 in AlexNet, presumably the run-time advantage would have been even better.

10 The "Broken Clock Problem"

Often a machine learning algorithm, though convergent, will produce the same class prediction for every case, the so-called "broken clock" problem. Remedying such a problem can be challenging.

10.1 Example: CIFAR-10 Data

We encountered "broken clock" with the famous CIFAR-10 dataset, which consists of 60,000 color images, each 32x32x3 in size, with ten classes.

Since there were 10 classes and the predictions were identical, the accuracy for pure liquidSVM was 10%. Here **TDAsweep** succeeded in avoiding the broken clock phenomenon. Of course, one could also avoid such an issue by performing further hyperparameter tuning, and 46.86% is still not a very good accuracy level for this dataset. Nonetheless, this simple example proves yet another possible advantage of **TDAsweep**: Dealing with convergence issues.

Table 4: **Example of TDAsweep against the broken clock problem.** Table demonstrating TDAsweep resolving the SVM broken clock phenomenon on the CIFAR-10 image set

	MNIST (raw)	**MNIST (sweep)**
Model Package	liquidSVM	TDAsweep+liquidSVM
No. of features	3072	1140
intervalWidth	NA	1
thresholds	NA	(25,100)
Accuracy (Validation Set)	10%	46.86%

.

11 Other Experiments

Table 5 shows the results of SVM trained on the various datasets without TDAsweep. Note that the e1071 version was used for Fashion-MNIST, and we used the liquidSVM version for the rest. Table 6 shows the results of SVM trained on the various datasets with TDAsweep.

Table 5: **Performance of pure SVM on various image data.** Results of SVM on Fashion-MNIST [10], kuzushiji-MNIST [11], histology-MNIST (28x28 image version), and EMNIST [12].

	MNIST (raw)	**MNIST (sweep)**	**MNIST (sweep)**	**MNIST (sweep)**
Model Package	e1071	liquidSVM	liquidSVM	liquidSVM
Sample size	70000	70000	5000	118000
No. of features	784	784	784	784
SVM train time	153 min	11.45 min	0.429 min	105.7 min
Accuracy (Validation Set)	89.6%	87.59%	62.9%	82.61%

Table 6: **Performance of TDAsweep+SVM on vaious image data.** Results of TDAsweep + SVM on Fashion-MNIST, kuzushiji-MNIST, histology-MNIST (28x28 image version), and EMNIST.

	MNIST (raw)	**MNIST (sweep)**
Model Package	liquidSVM	TDAsweep+liquidSVM
No. of features	3072	1140
intervalWidth	NA	1
thresholds	NA	(25,100)
Accuracy (Validation Set)	10%	46.86%

.

Again, we see that TDAsweep has an advantage on overall run-time (TDAsweep + SVM train) if the traditional **e1071** package is used to train, and we could also observe that TDAsweep could lose that speed advantage when **liquidSVM** was used. We would like to emphasize that advantage of run-time that TDAsweep brings scales with the complexity of the dataset and the algorithm used to train, as implied by the experiment results. Moreover, even with the smaller datasets and the faster evaluation of training models, our parallelization feature would still accelerate the process and provide the user with other merits such as a superb alternative for data augmentation, a potential cure for the broken clock problem, or simply to avoid overfitting. We believe that the scalability of TDAsweep will be especially usefull in today's research, where larger datasets and more complex models are emerging as the norms. We firmly believe that the multifaceted features TDAsweep have to provide will make TDAsweep a tremendous asset to the Machine Learning research community.

12 Acknowledgments

We thank Richard M. Levenson and his lab for providing us useful suggestions and medical datasets for testing; We thank Robin Yancey for her guidance on implementing the parallelization feature for TDAsweep.

References

[1] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks In *Proceedings of the 25th International Conference on Neural Information Processing Systems -*

Volume 1, NIPS'12, 2014 14th International Conference on, pages 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[2] Larry Wasserman Topological data analysis In *Annual Review of Statistics and Its Application, 5(1):501-532.*, 2018

[3] Chollet, François, and J. J. Allaire Deep learning with R In *Manning Publications*, 2018

[4] Kathryn Garside, Robin Henderson, Irina Makarenko, and Cristina Masoller Topological data analysis of highresolution diabetic retinopathy images In *Plos One, 14(5)*, 2019.

[5] Jing Dong, Wei Wang, Tieniu Tan, and Y.Q. Shi. Run-length and edge statistics based approach for image splicing detection In pages 76–87, 01 , 2008

[6] A.h. Mir, M. Hanmandlu, and S.n. Tandon Texture analysis of ct images In *IEEE Engineering in Medicine andBiology Magazine, 14(6):781–786*, 1995.

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

[8] Kather, Jakob Nikolas, Cleo-Aron Weis, Francesco Bianconi, Susanne M. Melchers, Lothar R. Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Zöllner Multi-Class Texture Analysis in Colorectal Cancer Histology In *Scientific Reports 6 (1)*, 2016.

[9] Alex Krizhevsky Learning multiple layers of features from tiny images In , 2009

[10] Han Xiao and Kashif Rasul and Roland Vollgraf Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms In *arXiv*, 2017-08-28

[11] Tarin Clanuwat and Mikel Bober-Irizar and Asanobu Kitamoto and Alex Lamb and Kazuaki Yamamoto and David Ha Deep Learning for Classical Japanese Literature In *arXiv*, 2018

[12] Cohen, Gregory and Afshar, Saeed and Tapson, Jonathan and Schaik, Andre Van EMNIST: Extending MNIST to handwritten letters In *International Joint Conference on Neural Networks (IJCNN), 2017*