# Topological Data Analysis of Convolutional Neural Networks' Weights on Images

Rickard Brüel Gabrielsson

**Abstract**

The topological properties of images have been studied for a variety of applications, such as classification, segmentation, and compression. For the application of image classification, high classification accuracy has been achieved by machine learning models, especially convolutional neural networks (CNNs), and we seek to understand the reasons behind their success. In our project, we apply topological data analysis to describe, visualize, and analyze topological properties of the weights learned from a CNN classifier trained on digit images from the MNIST data set.

## 1 Introduction

Machine learning for natural image classification is an important problem that has received a lot of attention recently and has been a major challenge for modern science and engineering. Machine learning models seek to replicate the remarkable ability of humans and many other animals to make sense of and classify natural images. Some insight into how was achieved by Hubel and Wiesel in the 1950s-1960s when they showed that the visual cortex of monkeys and cats contains neurons that individually respond to small regions of the visual field. Mirroring this, convolutional neural networks (CNNs), with tremendous success, assign weights to small regions (filters) of the pixels of an image instead of assigning individual weights to each pixel input as in a traditional fully-connected neural network. Over the course of training, CNN image classifiers update their weights using back-propagation to improve classification accuracy on the training dataset. However, multi-layer neural networks are less interpretable than simpler linear models, so how the high-level features learned by neural networks contribute to their success is not fully understood. In the endeavor to gain insight into what features of small image patches CNNs react to or learn and how the CNN weights evolve over training, network analysis gives us some promising techniques to apply to this problem that have never been used before on the weights of CNNs trained on images. Topological data analysis (TDA) has been applied to study natural image statistics and to generate dimensionality-reduced topological networks from data, since natural images provide rich structures within a high-dimensional point cloud where topological properties are far from obvious. By applying topological data analysis to the new context of learned weights of a CNN image classifier, we hope to enrich TDA as well as machine learning.

## 2 Related Work

As a response to the growth of data in a great variety of forms in both modern science and engineering, Carlsson [1] and [2] seeks to introduce topology as a helpful instrument with useful properties for coping with as well as understanding data. Data that are available to us today are often very high-dimensional, noisy, and in coordinates that have no clear interpretation. Methods from geometry and topology which study finite sets of points equipped with a metric, i.e. point clouds, are thus natural tools to consider. Topology, Carlsson argues, deals with qualitative geometric information, such as connectivity information (classifying loops as well as higher dimensional surfaces) and concerns geometric properties that are coordinate independent, representing the intrinsic geometric properties of the objects. Furthermore, through the process of simplicial approximation, most information about topological spaces can be obtained through diagrams of discrete sets. Topological data analysis (TDA) aims to create graphs (simplicial complexes) from data while preserving topological properties, i.e. properties that do not require domain expertise but are rather manifest in the data. Carlsson created the **Mapper Method** (which we describe in Section 3.2) to obtain qualitative understanding of high-dimensional point clouds via the generation of a representative graph.

Also, previous work by Lee et al. [4] analyzed high-dimensional point clouds from natural images and observed that the probability density of data points $\boldsymbol{x} \in \mathbb{R}^9$ (log intensity values for high-contrast image patches of size $3 \times 3$ pixels) is highest in clusters and non-linear low-dimensional manifolds. They focused on the 20% of image patches with highest contrast according to the $D$-norm measure of contrast defined by $\|\boldsymbol{x}\|_D = \sqrt{\boldsymbol{x}^T D \boldsymbol{x}}$, where $D$ is a particular symmetric positive definite $9 \times 9$ matrix. Preprocessing included dimensionality reduction via centering each patch to have zero mean, as well as contrast-normalization to whiten the data, resulting in a dataset of $4.5 \times 10^6$ points on the 7-dimensional sphere $S^7 \subset \mathbb{R}^8$. They discovered that the majority of high-contrast optical patches in this state space were concentrated around a non-linear continuous 2-dimensional submanifold within $S^7$, resembling blurred step edges parametrized by orientation angle $\alpha$ and displacement $l$ from the origin.

Both Lee et al and Carlsson et al have used methods from topology aimed at analyzing local structure of images by studying small patches, and found some significant properties. Since the first convolutional layer in a CNN will assign weights to small patches of an image, we should expect the same properties to be found in these weights, and even to a greater extent because CNNs are supposed to learn the *essence* of these patches in order to perform well on the image classification task. Not only does this provide a great empirical example of the application of Carlsson's topological methods, providing an example of success or weakness, but it could also potentially bring insight into the actual workings of CNNs in general. We build on prior work by showing how the Mapper method helps generate networks that distinguish poorly trained CNN weights from well-trained CNN weights, and apply network analysis on top of these networks.

## 3 Formal Mathematical Background

### 3.1 Density filtrations

To determine the 'core' subset of a set of points $X$, we perform a density filtration of the points based on a nearest neighbor estimate of the local density, which we define as follows. For $x \in X$ and $k > 0$, let $\rho_k(x)$ denote the distance between $x$ and its $k$-th nearest neighbor. $\rho_k$ is thus inversely proportional to the density at $x$, so we define the density as its inverse. A core subset $X(k, p)$ is then defined as the $p$ percent of points with the highest $\rho_k$ density, and this subset will serve to approximate the space $X$.

### 3.2 Mapper method

We will now describe Carlsson's [1] Mapper method. Letting $X$ be any topological space and $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ any covering of $X$, Carlsson defines the *nerve* of $\mathcal{U}$ (denoted $N\mathcal{U}$) as the abstract simplicial complex with vertex set $A$, and where a family $\{\alpha_0, \ldots, \alpha_k\}$ spans a $k$-simplex if and only if $U_{\alpha_0} \cap \cdots \cap U_{\alpha_k} \neq \emptyset$. The simplicial complex $\check{C}^{\pi_0}$ is then defined to be the nerve of the covering of $X$ by sets which are path connected components of a set of the form $U_\alpha$. Carlsson uses the single linkage clustering algorithm, which is defined by fixing a value of a parameter $\epsilon$, and defining a relation $R_\epsilon$ over $X$ by $(x, x') \in R_\epsilon$ if and only if $d(x, x') \leq \epsilon$. The point cloud is then partitioned by the set of equivalence classes under the transitive closure $\sim_\epsilon$ of $R_\epsilon$. This allows us to make sense of the notion of $\pi_0$, that is, how to construct connected components of a point cloud. In this context, the Mapper method allows us to construct $\check{C}^{\pi_0}$ in the following way:

1. Define a reference map $f : X \to Z$, where $X$ is the given point cloud and $Z$ is the reference metric space.
2. Select a covering $\mathcal{U}$ of $Z$.
3. If $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, then construct the subsets $X_\alpha = f^{-1}U_\alpha$.
4. For the single linkage clustering algorithm above, select a value $\epsilon$ as input. Apply the single linkage clustering algorithm with parameter $\epsilon$ to the sets $X_\alpha$ to obtain a set of clusters. This gives us a covering of $X$ parametrized by pairs $(\alpha, c)$, where $\alpha \in A$ and where $c$ is one of the clusters of $X_\alpha$.
5. Lastly, construct the simplicial complex whose vertex set is the set of all possible such pairs $(\alpha, c)$ and where a family $\{(\alpha_0, c_0), \ldots, (\alpha_k, c_k)\}$ spans a $k$-simplex if and only if the corresponding clusters have a point in common.

These procedures give us a simplicial complex (a network) from which we will be able to discern qualitative properties of the point cloud.

If $Z = \mathbb{R}^n$ in the Mapper (as it is for our project) then a covering can be obtained in the following way: Let $X = \mathbb{R}$, and let $R$ and $e$ be positive real numbers. Then construct the covering $\mathcal{U}[R, e]$ of $X$ by all intervals of the form $[kR - e, (k+1)R + e]$, which is a two parameter family of coverings, and as long as $e < \frac{R}{2}$ it has covering dimension 1 in the sense that no non-trivial threefold overlaps are non-empty. In order to obtain a covering of $\mathbb{R}^n$ we may then use products of these intervals.

## 4 Data Collection Procedure

To generate a dataset of learned weights, we trained a multilayer convolutional neural network (CNN) on the MNIST [3] dataset of hand-written digits. The MNIST dataset consists of images with size 28 pixels by 28 pixels, where each pixel is a greyscale intensity between 0 and 1. Each image is associated with a label from 0-9 of the correct classification of the digit.

We used the following layers:

1. Convolutional Layer with 512 hidden nodes and $5 \times 5$ filters (weights) with stride 1, followed by RELU nonlinearity and max-pooling with stride 2
2. Another Convolutional Layer with 128 hidden nodes and $5 \times 5$ filters (weights) with stride 1, followed by RELU nonlinearity and max-pooling with stride 2
3. Densely Connected Layer with 1024 hidden nodes
4. Dropout with dropout rate 0.5
5. Readout Layer using softmax activation and cross-entropy loss function

After training our CNN for 1000 iterations with batch size 50, we achieved an accuracy of approximately 96% on the held out MNIST test set of 10,000 images. Since each of the 512 hidden nodes in the first convolutional layer has its own $5 \times 5$ filter, we obtain 512 weight matrices of size $5 \times 5$. Thus, we obtain a point cloud of 512 points $w \in \mathbb{R}^{25}$.

Next, we preprocessed these first-layer weights by mean-centering and scaling each of the 512 weight vectors $w$ to have mean 0 and variance 1. Then we applied the $k$-nearest neighbor Density filtration function described formally in Section 3.1 above with $k = 5$ and percentage $p = 20\%$, giving us $\lfloor 0.2 \cdot 512 \rfloor = 102$ data points $w' \in \mathbb{R}^{25}$.

After a lot of experimentation, we chose the following model as focus for our project to generate networks from our preprocessed dataset of learned weights. **Model 1**: Neighborhood Lenses 1 and 2 as reference maps with Variance Normalized Euclidean distance metric.

Furthermore, a *lens* (as termed by the Ayasdi software) is a function which works as the reference map $f : X \to Z$ described in Section 3.2. A lens converts a dataset $X$ to a vector $z \in \mathbb{R}^n$, where each row in the original dataset contributes to a real number in the vector. Essentially, a lens operation turns every row into a single number. When $m$ different lenses are applied to a data point, the result is the Cartesian product of the output of each lens. For example, two lenses map each data point in the original dataset $X$ into the two dimensional open covered space $\mathbb{R}^2$. The lens(es) determine how the data is partitioned into bins for the clustering step of the Mapper method, and hence can bring various aspects of the data into focus.

Together, Neighborhood Lens 1 and Neighborhood Lens 2 generate an embedding of high-dimensional data points into $\mathbb{R}^2$. A $k$-nearest neighbors graph of the data is generated by connecting each point to its nearest neighbors using a provided distance metric and the Ayasdi software embeds the graph in two dimensions. The first lens outputs the $x$-coordinate of the embedding, and the second lens outputs the $y$-coordinate. These lenses work to emphasize the metric structure of the data.

The computation of the lens function depends on the notion of distance between points in the dataset. We experimented with many distance metrics, but found the Variance Normalized Euclidean distance metric to be the most appropriate and informative for our dataset.

Variance Normalized Euclidean distance metric ($d_{\text{VNE}}$) is a variant of standard Euclidean distance that first standardizes the distance for each column (in our case, weight variable) of the dataset by dividing by its variance. Hence this metric is appropriate in the scenario that the columns in the data set could have significantly different variance. Formally, the Variance Normalized Euclidean distance between two data points $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ in dataset $X$ is defined by:

$$d_{\text{VNE}}(x, y) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{v_i}} \qquad \text{where} \qquad v_i = \frac{1}{m}\sum_{j=1}^{m}(X_{j,i} - \hat{\mu_i})^2 \qquad \text{where} \qquad \hat{\mu_i} = \frac{1}{m}\sum_{j=1}^{m} X_{j,i}$$

where $v_i$ denotes the variance associated with column $i$, and $m$ denotes the number of rows in $X$, and $\hat{\mu}_i$ denotes the mean of column $i$ in dataset $X$.

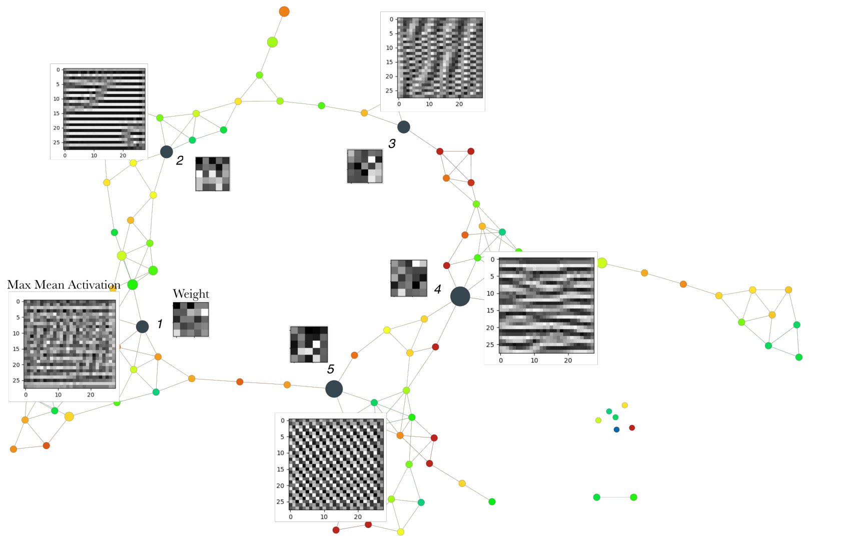
## 5 Preliminary Results

We used the Ayasdi software to implement the Mapper method described in Section 3.2 and generate networks from a variety of models. Here, we include our initial findings from model chosen in Section 4.

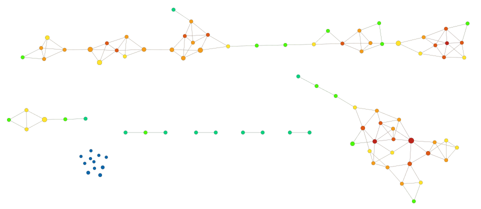
While generating networks from models on our CNN's first-layer weights after training, we compared the topological properties of the first-layer weights prior to and after training. The network that differed the most prior and post training was a network using Neighborhood Lenses 1 and 2 along with the Variance Normalized Euclidean Metric. We normalized and preprocessed the weights (as described in Section 4.2) by density filtration to obtain 102 weights. The nodes in the resulting networks generated from these weights essentially correspond to *supernodes* (clusters of weights) due to the clustering algorithm used by the Mapper method. The weights prior to training were randomly initialized according to a Gaussian distribution with mean 0 and variance 0.1. We see from Figure 1b that *before* training, the weights seem to lack coherent structure and the generated network consists of several separate connected components. We see that *after* (Figure 1a) training, the weights seem to form into more of a clear network structure: a **ring**.

In prior analysis of image patches, Carlsson et al [2] also found the topological property of a circle (or ring) in their data. They conjectured that this circle had the property that as one traveled along it, the image patch data changed from vertical edges to diagonal edges, and then from diagonal to horizontal edges, and so on (see Figure 1c). In Figure 1a we can see five nodes taken along the circle in the trained network. While it is fairly hard to conjecture the vertical, diagonal, or horizontal property of a weight directly, we see that if we compute and plot the input to the weights within a node that maximizes the mean (or sum) of the corresponding neurons' activations (channel), then these properties become much clearer. For example, at node 2 we see a horizontal pattern and we see the patterns turn along the circle to a more diagonal pattern at node 3 and then to, once again, a fairly horizontal pattern at node 4. At node 5 again we see a clear diagonal pattern that evolves continuously via node 1 to a vertical pattern in node 2.

From this, we conjectured that path length along this circle measures the rotation of this pattern, as also conjectured by Carlsson et al [2]. Next we tried expanded our results from Figure 1a to include all nodes along the circle for close inspection of the corresponding weights and max-activations (Figure 2). The rotation of the slope is non-obvious, but Figure 2 does suggest that nodes close have a similar slope that somewhat smoothly rotates along the circle.



(a) After training: Nodes with corresponding weights visualized with its max mean activation input


(b) Before training: Gaussian, mean 0, stddev: 0.1

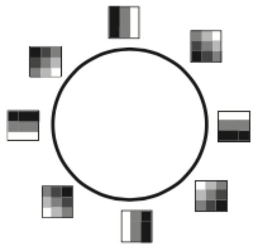

(c) Circle of image patches from prior work by Carlsson et al [2]

Figure 1: Comparison of maximally activating patches from CNN and images patches from Carlsson et al [2]

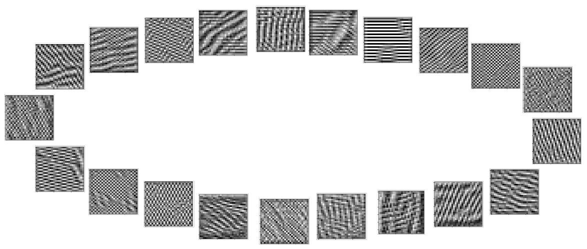


Figure 2: Max activation along circle in Model 1

From these preliminary results, and from the results obtained from less trained weights, our main hypothesis is that weights optimize to capture all lines and edges. I.e. it optimizes to detect straight lines of any rotation. If this is true, the weights would learn to detect edges of different rotations and as it becomes able to detect enough angles from 0 to $2\pi$ (alternatively from 0 to $\pi$) it at the same time naturally starts to approximate the circular topological property. Since to do well, it has to spread it's rotation-detection evenly around a full rotation, which allows us to travel around the circle smoothly as in Figure 2. Thus, as the weights improve and are learning more of the true representation of the input data, the approximated rotations becomes smoother, which allows the weights to classify slopes more exact and for the whole CNN to perform better. Experiments below with synthetic data seem to support this hypothesis.

## 6 Synthetic Data

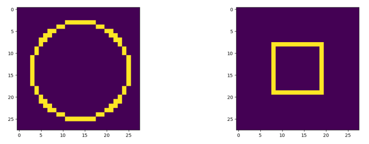
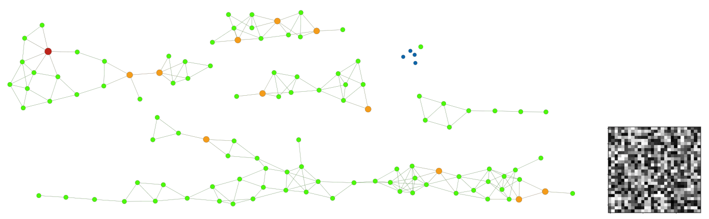
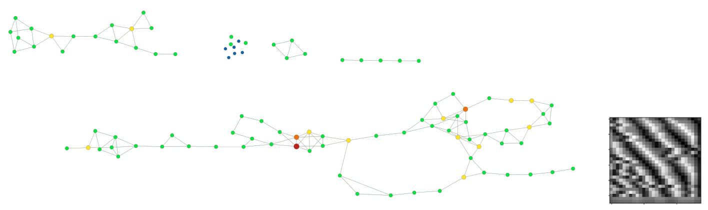


Figure 3: Synthetic circle and square

Investigating the effects of different image data on the weights learned and the resulting network, we generated and trained the same CNN on synthetic data (Figure 3). The data consisted of $28 \times 28$ pixel images with either a circle or a square of random radius and width respectively. Then the CNN was trained to classify the image as a circle or a square, a much easier task than MNIST digit classification. We generated two different networks (according to Model 1) based on the synthetic data (10,000 training data, 2,500 test data), one after 40 batch iterations and where the network was achieving 100% accuracy, and another after 2,000 batch iterations where the network had achieved a training loss of order $10^{-8}$.


(a) Network and max-activation after 40 batch iterations


(b) Network and max-activation after 2000 batch iterations

The network generated (Figure 4a) after 40 batch iterations does not display any interesting unifying topological structure, but is fragmented into separate connected components. Similarly, the weights and corresponding max-activation are noisy and have not changed significantly since their random initialization. On the other hand, the network generated (Figure 4b) after 2,000 batch iterations has started to display some interesting topological properties, with one major circle forming as well as some smaller circles, and a major flare on the periphery. By inspecting the weights and their corresponding max-activation, we see that they now exhibit straight edges with different slopes, a pattern also found among our weights trained on the MNIST data.

Unsurprisingly, this suggests that in some tasks where classification is too easy, it doesn't seem necessary to learn all of the true structure of the data. However, minimizing an appropriate cost function can still cause the weights to learn to better approximate linear patterns. Even though the data sources are different, training the CNN on the synthetic data leads to similar results as training it on MNIST, suggesting that the topological properties generalizes over these two datasets.

## 7 An Improved Model

Since our preliminary results were promising but still very noisy (both the graphs and the weight-visualization) we explored ways to achieve more exact results. We trained countless of CNNs on the MNIST dataset on Google Cloud GPU. We realized that many of our weights were not changing (learning) very much from their initialization, as also found by Shang et al [5]. The best results was achieved by a CNN with a similar configuration as before but now instead with only 64 hidden nodes in the first convolutional layer, 8 hidden nodes in the second convolutional layer, and 512 hidden nodes in the fully connected layer. We trained this model a 100 times with batch size 100 for 10000 iterations till about .990 test accuracy. We tried filtration of the weight in terms of different norms and standard deviation to get the weights that had *learned* the most. Best results were achieved with the original density filtration which gave us 1280 weights. We then created a graph using the same metric as before but now with two PCA lenses. The result can be seen in Figure 5.

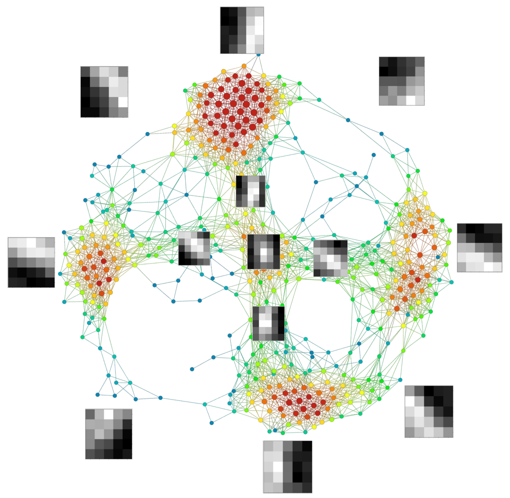


Figure 5: Network from improved model and corresponding averaged weights

This is strikingly similar to the circle that was hypothesized in section 5. Also with more data points, we could now take averages of weights close to each other in the graph instead of looking at max activations. This gives us a clear picture over the topological structure of the learned weights. It is also easy to see that the topological circle is not more protruding due to the learned overlap (i.e. edges on both sides of the weight) that can be seen in the four inner weight-averages in the figure. Since this is the case, we get one big circle and several sub-circles. Thus, instead of a first order approximation of the Klein bottle formed by patches as conjectured by Carlsson et al [2] in form of the circle, we are finding something that is much closer to the actual Klein bottle and Three Circle Model (see figure 6) as found by Carlsson et al [2]. We believe the improved results came from reduced noise from more data, and fewer first layer weights and also fewer nodes in other hidden layers led to increased learned structure of the first layer weights.

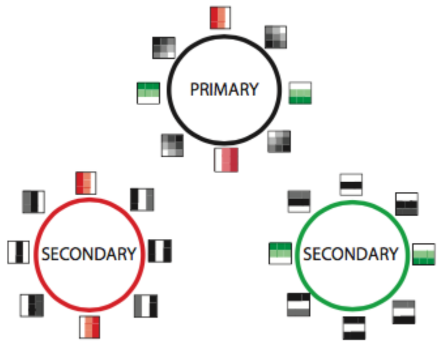


Figure 6: Three circle model in the data from Carlsson et all [2]

## 8 Overall Analysis

Whilst the space of natural images has a complicated topology, it can be better understood through an approximation by homology groups $H_k$, where the first $k$ groups form a $k$-th order approximation to the topology. By normalizing our data, the zeroth-order terms $H_0$ in the approximation are eliminated. The first-order term $H_1$ then encompasses the most fundamental structure of natural images, which is linear gradients. Since linear gradients are canonically parametrized by their angle, which is naturally encompassed by a circular structure, the first order approximation to our topology is a circle. Carlssons's work actually shows that the first Betti number is three, indicating that there are three generators for $H_1$. The linear gradients form the primary generator, but there are two secondary generators that arise from quadratic gradients. In our paper, we first consider the primary generator and its corresponding linear gradients. However, ultimately we found something that is much closer to encompassing all the three generators.

## 9 Conclusion

Through our project, we extended topological data analysis to the weights of a CNN. This problem was difficult because we have no prior work to refer to regarding networks being built from the weights of a CNN. Hence, we spent a large amount of time finding and experimenting with appropriate and interesting model for generating our network. We also had to learn how to use the Ayasdi software, how to visualize images that maximally activate the CNN's weights, and to use the Google Cloud platform to train a large number of networks. Moreover, since our project is naturally more qualitative and the training procedure has underlying noise, interpreting our results was a challenge. Nevertheless, by applying TDA to a new context of CNN weights, we have contributed to the advance of TDA and machine learning.

## 10 Contributions and Acknowledgements

This project is done under the partial supervision of Stanford Professor Gunnar Carlsson, whose prior work is cited under our related work section. This project has shared components with a project in CS224W with Heather Blundell and Dylan Liu.

## References

[1] Carlsson, G. Topology and data. *Bulletin of the American Mathematical Society 46*, 2 (2009), 255–308.

[2] Carlsson, G., Ishkhanov, T., Silva, V. D., and Zomorodian, A. On the local behavior of spaces of natural images. *International Journal of Computer Vision 76*, 1 (2007), 1–12.

[3] LeCun, Y., and Cortes, C. MNIST handwritten digit database.

[4] Lee, A. B., Pedersen, K. S., and Mumford, D. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision 54*, 1 (Aug 2003), 83–103.

[5] Shang, W., Sohn, K., Almeida, D., and Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units. *CoRR abs/1603.05201* (2016).